TARGET CLASSIFICATION UNDER MULTI SENSOR ENVIRONMENT

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

BY

BENGÜ ATICI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

TARGET CLASSIFICATION UNDER MULTI SENSOR ENVIRONMENT

submitted by **BENGÜ ATICI** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

Prof. Dr. Esra Karasakal
Supervisor, **Industrial Engineering, METU** _____

Assoc. Prof. Dr. Orhan Karasakal
Co-Supervisor, **Industrial Engineering, Çankaya University** _____

Examining Committee Members:

Prof. Dr. Ömer Kırca
Industrial Engineering, METU _____

Prof. Dr. Esra Karasakal
Industrial Engineering, METU _____

Prof. Dr. Sinan Gürel
Industrial Engineering, METU _____

Assoc. Prof. Dr. Seçil Şavaşaneril
Industrial Engineering, METU _____

Assist. Prof. Dr. Haluk Aygüneş
Industrial Engineering, Çankaya University _____

Date: 09.09.2019

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: Bengü Atıcı

Signature:

ABSTRACT

TARGET CLASSIFICATION UNDER MULTI SENSOR ENVIRONMENT

Atıcı, Bengü
Master of Science, Industrial Engineering
Supervisor: Prof. Dr. Esra Karasakal
Co-Supervisor: Assoc. Prof. Dr. Orhan Karasakal

September 2019, 166 pages

Radar systems have important roles in both military and civilian applications. As the capabilities increase in terms of range, sensitivity and the number of tracks to be handled, the requirement for Automatic Target Recognition (ATR) increases. ATR systems are used as decision support systems to classify the potential targets in military applications. These systems are composed of four phases, which are selection of sensors, preprocessing of radar data, feature extraction and selection, and processing of features to classify potential targets. In this study, we focus on the classification phase of an ATR system having heterogeneous sensors and develop a novel multiple criteria classification method based on modified Dempster-Shafer theory. Ensemble of classifiers is used as the first step probabilistic classification algorithm. It is treated as the state of the art technology for classification in which each single classifier is trained separately, and then the results of them are combined through several fusion algorithms. Artificial Neural Network and Support Vector Machine are employed in ensemble. Each non-imaginary dataset coming from multiple heterogeneous sensors is classified by both classifiers in the ensemble, and the classification result that has higher accuracy ratio is chosen for each of the sensor. After getting probabilistic classification of targets by different sensors, modified

Dempster-Shafer data fusion algorithm is used to combine the sensors' results to reach the final classification of targets.

Keywords: Artificial Neural Network, Support Vector Machine, Ensemble of Classifiers, Data Fusion, Dempster-Shafer Theory

ÖZ

ÇOKLU SENSÖR ORTAMINDA HEDEF SINIFLANDIRMA

Atıcı, Bengü
Yüksek Lisans, Endüstri Mühendisliği
Tez Danışmanı: Prof. Dr. Esra Karasakal
Ortak Tez Danışmanı: Doç. Dr. Orhan Karasakal

Eylül 2019, 166 sayfa

Radar sistemlerinin hem askeri hem de sivil uygulamalarda önemli rolleri bulunmaktadır. Menzil, hassasiyet ve takip edebilecekleri hedef sayısı arttıkça, Otomatik Hedef Tanıma (ATR) sistemlerinin gereksinimi de artmaktadır. ATR sistemleri, askeri uygulamalardaki potansiyel hedefleri sınıflandırmak için karar destek sistemleri olarak kullanılmaktadır. Bu sistemler, sensör seçimi, radar verilerinin ön işlemesi, özellik çıkarımı/seçimi ve potansiyel hedeflerin sınıflandırılması olarak dört aşamadan oluşmaktadır. Bu çalışmada, ATR'nin sınıflandırma aşamasına odaklanarak değiştirilmiş Dempster-Shafer veri birleştirme algoritmasına dayanan çok kriterli hedef sınıflandırma yöntemi önermekteyiz. Çalışmamızda sınıflandırıcı topluluğu (Ensemble of Classifiers) sınıflandırma algoritması olarak kullanılmaktadır. Her bir sınıflandırıcının ayrı ayrı eğitildiği sınıflandırıcı topluluğu ile olasılıksal sınıflandırma yapıldıktan sonra sensörlerden gelen veriler füzyon algoritması ile birleştirilmektedir. Önerdiğimiz sınıflandırıcı topluluğunda Yapay Sinir Ağı ve Destek Vektör Makinesi, olasılıksal sınıflandırıcılar olarak kullanılmaktadır. Birden fazla heterojen sensörden gelen her veri kümesi, topluluktaki her iki sınıflandırıcı tarafından sınıflandırıldıktan sonra sensör veri kümelerinin her biri için daha yüksek doğruluk oranına sahip olan sınıflandırıcının sonucu seçilmektedir. Hedeflere ait verilerin farklı sensörler tarafından olasılıksal bir

şekilde sınıflandırılmasının ardından değiştirilmiş Dempster-Shafer veri birleştirme algoritması, sensörlerin sonuçlarını birleştirerek hedeflerin nihai sınıf atamasını yapmaktadır.

Anahtar Kelimeler: Yapay Sinir Ağı, Destek Vektör Makinesi, Sınıflandırıcı Topluluğu, Veri Füzyonu, Dempster-Shafer Teorisi

To my family…

ACKNOWLEDGEMENTS

First of all, I would like to thank to my supervisors Prof. Dr. Esra Karasakal and Assoc. Prof. Dr. Orhan Karasakal for their guidance, support and encouragements throughout this study. Thanks to them, I had a great chance to learn so many things.

Secondly, I would like to thank to my examining committee members Prof. Dr. Ömer Kırca, Prof. Dr. Sinan Gürel, Assoc. Prof. Dr. Seçil Savaşaneril and Assist. Prof. Dr. Haluk Aygüneş for their valuable time that spend for my work and for their valuable comments.

I owe my deepest thanks to my mother Fatma Atıcı, my father Ali Paşa Atıcı and my sister Bilge Atıcı Şevketbeyoğlu, who brought me to these days, never give up their endless support and patience to me, and for their unconditional love. I know that without their encouragements and support, I could not finish my master study.

Also, I would like to thank to my big family members for still loving me even though I could not participate any activity during my whole master study.

I also would like to thank to my first car, İbiş, who sacrificed himself for me in my car accident. Without him, I could not participate any of my thesis meetings.

Finally, I would like to thank to a very special person to me, Eren Güney, for his patience throughout my thesis study. I am also grateful to him for his love, support and encouragements through the last five years.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

ATR	Automatic target recognition
SVM	Support vector machine
ANN	Artificial neural network
DST	Dempster-Shafer theory
AI	Artificial intelligence
ML	Machine learning
MCDM	Multiple Criteria Decision Making
QP	Quadratic programming
SMO	Sequential minimal optimization
MSE	Mean square error
FOD	Frame of discernment
BPA	Basic probability assignment
ECDST	Ensemble of Classifiers with Classical Dempster-Shafer Theory
ECY	Ensemble of Classifiers with Yong et al. (2004)
ECMDF	Ensemble of Classifiers with Modified Distance Function
ECMDFS	Ensemble of Classifiers with Modified Distance Function and Sensor Accuracy
h-ECMDF	Hybrid Ensemble of Classifiers with Modified Distance Function
h-ECMDFS	Hybrid Ensemble of Classifiers with Modified Distance Function and Sensor Accuracy
PCP	Percent of correct predictions

AC	Average probability change in all targets
CC	Average probability change in correctly predicted targets
WC	Average probability change in wrong predictions

CHAPTER 1

INTRODUCTION

Military power is the whole of the resources comprises physical, technological, psychological and political power that a country uses to implement its national policies, and achieve its goals. It is very difficult to protect the existence of a country and integrity of its nation without military power. Thus, there has been increasing research and study to improve the military technology and power all over the world. According to the study which was conducted by Tian et al. (2017), world has spent a total of $1.74 trillion on military in 2017.

Radar systems have important roles both in military and in civilian applications. Those systems changed the way armies have fought since their invention during the World War II. As capabilities of those systems in terms of range, sensitivity and number of tracks that can be followed increase, popularity of automatic target recognition (ATR) systems has also increased. ATR systems take readings from radars, and then process it to recognize the class of targets. Based on the number of data sources used, ATR systems can be divided into two categories, which are single data source and multiple data source. Contrary to single data source systems, multiple data source systems can provide complementary knowledge about the target, and hence greater accuracy.

ATR systems are composed of four phases, which are selection of data sources, preprocessing/segmentation of data collected, feature selection and extraction, and classification (Roger et al., 1995). In the first phase, type, number, and physical locations of the data sources are decided depending on the types of the targets that the system will classify. Afterwards, preprocessing/segmentation of the data is conducted. In this phase, raw data is processed by data mining techniques, and relevant parts of it is determined by segmentation. In the third stage, informative features are selected.

Finally, classification algorithms are employed to identify the targets in the last phase. Figure 1.1 depicts the processes of ATR systems.

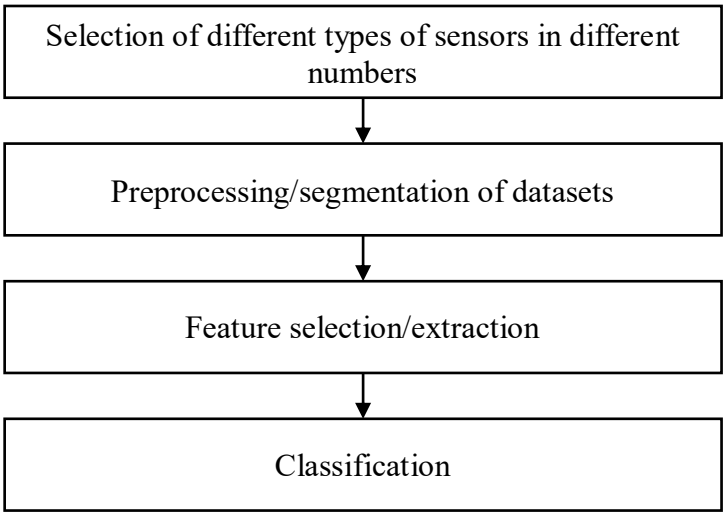

Figure 1.1. Processes of ATR system (Rogers et al., 1995)

According to the measurements produced by sensors, different types and different number of sensors may be used as data sources such as ultra-high range-resolution radar profiles, synthetic aperture radar and laser radar (Rogers et al., 1995) in ATR systems. Thus, sensor readings in ATR systems are usually heterogeneous.

In this thesis, classification phase of ATR systems is studied. According to the Sagi and Rokach (2018), errors of the individual classifiers are compensated by the combination of several classifiers' results, which makes the classification algorithm more robust to errors. Thus, ensemble of classifiers is used as a classification algorithm in our study. It is treated as the state of the art technology for classification in which each single classifier is trained separately, and then the results of them are combined through several fusion algorithms.

Consider a battlefield in which multiple heterogeneous sensor readings come for the same potential target. We want to identify the target as soon as possible before

engaging the target if necessary. To achieve that, after preprocessing the sensor readings and feature selection/segmentation, sensor readings must be classified. In multi-sensor target classification problem, ensemble of classifiers are employed. Artificial neural network (ANN) and support vector machine (SVM) are selected as individual classifiers in the ensemble due to their high classification power (Sagi and Rokach, 2018). As sensor readings have different features data, they cannot be fused directly. Thus, we firstly employ probabilistic classification for each sensor, and we fuse those probabilistic classification results by modified Dempster-Shafer Theory (DST). In the ensemble, ANN and SVM are trained for each sensor for each specific environment (i.e. dataset), and the classifier with higher accuracy is chosen.

The objective of the overall system is to predict the class of targets correctly with the largest possible probability and the maximum accuracy in the shortest time.

The main motivation of the study is to develop new methods that allow us to calculate the evidence distances in an elastic way in paradoxical and high conflicting situations in DST. The proposed way of calculating the distances are used in calculation of credibility degrees of the sensors. The proposed algorithm enables integration of machine learning (ML), Multiple Criteria Decision Making (MCDM), and DST and expands application of DST from single target to multiple targets.

The rest of the thesis is organized as follows: in Chapter 2, literature review on ANN, SVM and DST are given. In Chapter 3, we give proposed algorithms in detail. Chapter 4 introduces the benchmark models to assess the performance of the proposed algorithms. Details of the computational experiments and discussions are given in Chapter 5. Chapter 6 introduces the hybrid extension of the proposed algorithms, and provide the related computational results. Finally, Chapter 7 concludes the thesis.

CHAPTER 2

LITERATURE REVIEW

Classification problem deals with predicting the class of the observations. It is a well-known problem in ML. ML is a branch of artificial intelligence (AI). In the simplest form, ML gives ability to computers to learn, and perform certain operations without human intervention.

According to the learning algorithms, ML is classified into three groups, which are supervised, unsupervised and reinforcement learning. In supervised learning, the model learns from labeled dataset. The data fed to the model has input and output pairs. Supervised learning can be divided further as classification and regression. In classification, the model is trained to predict the category of the new observation based on predefined classes. Here, the output is qualitative. On the other hand, regression models are trained to predict a continuous value for the given input based on previous input and output relation.

In unsupervised learning, dataset has input values but it does not have related output values. The general unsupervised learning problem is clustering. By clustering, similar observations are grouped together by minimizing the differences between the groups, and maximizing the difference between the groups.

There is also another learning algorithm that is not either supervised or unsupervised. Reinforcement learning is based on trial and error. It aims to teach machine to act itself in dynamic environments by maximizing the reward signal.

Kotsiantis (2007) summarizes the most popular classification algorithms in ML as in Table 2.1, which are decision trees, artificial neural network, naïve Bayes, k-nearest neighbors, support vector machine and rule-learners.

Table 2.1. *Review of classification algorithms in ML (Kotsiantis, 2007)*

Attributes	**Decision Trees**	**ANN**	**Naïve Bayes**	**k-Nearest Neighbor**	**SVM**	**Rule-Learners**
Accuracy in general	**	***	*	**	****	**
Speed of learning	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes	**	***	*	*	***	**
Dealing with discrete/binary/continuous attributes	****	*** (not discrete)	*** (not continuous)	*** (not discrete)	** (not discrete)	*** (not continuous)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classification	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

* represents the lowest and **** represents the highest performances. The other ones are the intermediate values.

In ATR systems, the main aim is to identify the targets correctly and quickly so that required weapons can be directly engaged to them. Speed and classification accuracy may be regarded as the most critical objectives of ATR systems. In the last phase of the ATR, to classify the targets into classes, individual or combination of the classifiers are used. We may increase the classification power of ATR systems by employing ensemble of classifier. When the combination of classifiers is used, fusion algorithm to combine the outputs of the each classifier to get a single prediction from them is needed. From Table 2.1, we can see that ANN and SVM are the most prominent classifiers in accuracy in general and speed of classification. Hence, these two classification methods are further investigated in Section 2.1 and 2.2, respectively. In Section 2.3, ensemble of classifiers is reviewed. Finally, in Section 2.4, review of DST, which is an effective fusion method to combine separate pieces of evidences coming from different data sources under uncertainty and imprecision, is given.

2.1. Perceptron Based Techniques

In their article titled “A logical calculus of the ideas”, McCulloch and Pitts (1943) investigate that how human brains process complex patterns, and how this process can be represented by artificial neurons that are the building blocks of ANN. ANN is one of the most popular classification algorithms in ML. It has a wide range of application areas such as image processing, natural language processing, pattern recognition, emotion classifier. Also, regression and clustering can be performed with ANN.

To better understand it, we should first look at the structure of a single perceptron.

2.1.1. Single Layer Perceptron

Single layer perceptron is introduced by McCulloch and Pitts (1943). Their model is composed of several input nodes that takes the feature values of the input data, and output node where the recommendation of the perceptron is received as 0 and 1.

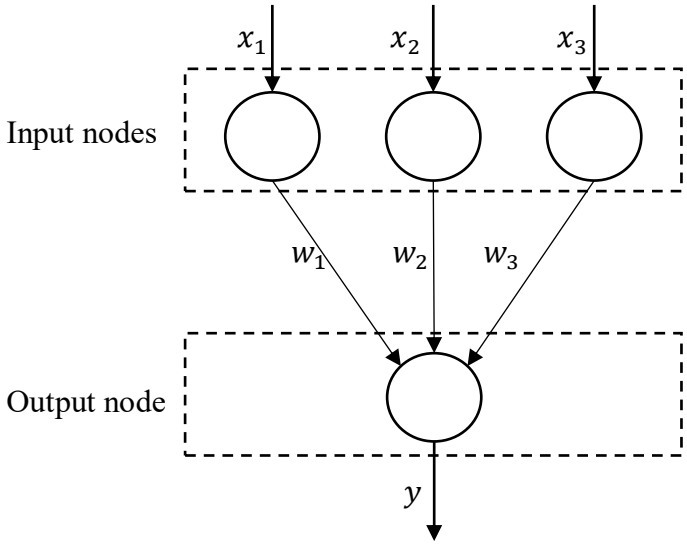

Figure 2.1. Structure of single layer perceptron

Figure 2.1 shows the topology of single layer perceptron. w_i and x_i represent the weights between input and output layers, and input values, respectively. y is the output of the perceptron.

Firstly, input values, x_i, are multiplied by the corresponding weights, w_i, and then weighted inputs, $w_i * x_i$, are summed together according to (2.1). If this sum is greater than the threshold, θ, then the signal is propagated to the output node, and 1 is received as the output of the perceptron. On the other hand, if the weighted sum is less than the threshold, then the perceptron receives 0 as the recommendation. However, McCulloch and Pitts' model (1943) is only applicable to linearly separable datasets.

$$y = \begin{cases} 1 & if \sum_{i=1}^{n} w_i * x_i > \theta \\ 0 & otherwise \end{cases} \tag{2.1}$$

2.1.2. Artificial Neural Network

Single layer perceptron is not suitable for nonlinear datasets. To overcome nonlinearity, multi-layer perceptron or ANN is developed. As opposed to single layer perceptron, ANN is composed of multiple layers. In addition to increased number of layers, multi-layer perceptron uses nonlinear activation function. Figure 2.2 shows the structure of ANN.

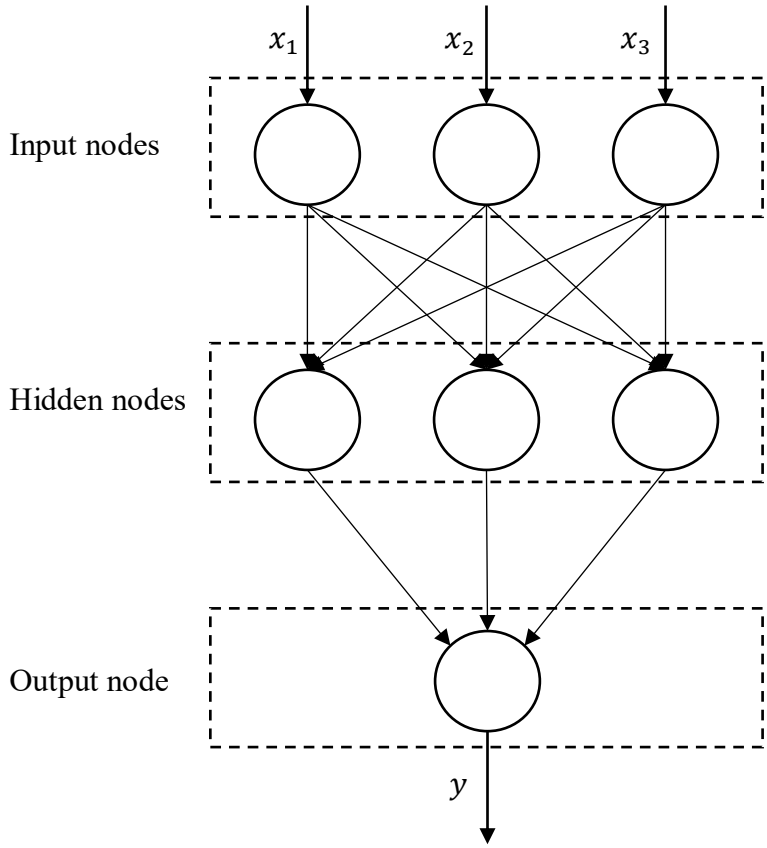

Figure 2.2. Structure of ANN

ANN is composed of input, hidden and output layers. There may be more than one hidden layer or not at all. Hidden layers connect the input and output layer. In each layer, there are certain numbers of nodes, which are called neurons. Every neuron

takes the weighted sum of the outputs of the previous layer as its input, and processes it according to the certain activation function as its output to the next layer.

Every connection between the nodes has weights. These weights adjust the performance of the network. Input vector is fed to the input layer. Then, according to the weights between input and hidden layer, weighted sum of the input vector is forwarded to the next layer, which is hidden layer in Figure 2.2. In hidden layer, weighted sum is processed according to predefined activation function. Next, this signal is forwarded to the last layer, which consist of output nodes. At the final stage, weighted sum is generally modified so that it is between 0 and 1 (Kotsiantis, 2007). In this case, modified weights represent probabilities, and ANN recommends the class of the output node that has the highest probability. Alternatively, threshold value may be used so that if the weighted sum forwarded to output nodes is lower than the threshold, 0 is received, and otherwise, 1 is received (Kotsiantis, 2007).

The aim of ANN is to adjust the weights so that the total error is minimized. This is achieved by training the network. The most popular training algorithm is backpropagation algorithm (Rumelhart et al., 1986). However, the deficiency of backpropagation is the speed of training. Random weights algorithm, which has attracted researches attention recently, is another training algorithm (Cao et al., 2018). Other than these, genetic algorithms (Siddique and Tokhi, 2001) and Bayesian methods (Vivarelli and Williams, 2001) are among the other alternative training algorithms.

Selection of activation function, network structure and weights of the connections are three aspects that significantly affect the overall performance of ANN (Kotsiantis et al., 2006). According to Géron (2018), there are three popular activation functions in terms of performance, which are rectified linear, hyperbolic tangent, and sigmoid activation function. Zhang et al. (2018) divide the most commonly used network architectures into four categories, which are auto encoders, deep belief networks, convolutional neural networks and recurrent neural networks. Once activation

function and network structure is determined, they do not change with the input or output values. Apart from them, weights are adjusted to minimize the total deviation from the target values, and that is how the ANN learns.

Determining the number of neurons in hidden layers is another aspect affecting the performance of ANN. If there are more than enough neurons in hidden layers, model may over fit the data. On the other hand, if the number of neurons is less than needed, the performance of ANN may be poor. In their article, Camargo and Yoneama (2001) give detailed discussion on this issue.

To sum up, there are many parameters that change the performance of ANN. Thus, there is no specific ANN type that suits any given problem. A detailed review on ANN can be found in Schmidhuber (2015).

2.2. Support Vector Machine

SVM is the most known type of kernel methods, which mainly deal with classification (Chollet, 2018). Vapnik and Chervonenkis (1963) introduced SVM as a linear formulation. Then, in 1995, Vapnik and Cortes (1995) proposed a nonlinear formulation of SVM with the concept of soft margin. By maximizing the margin between different classes, SVM tries to find the optimal separating hyperplane so that different types of instances are separated with maximum distance. To classify the new data points, the algorithm checks which side of the hyperplane that the observation falls on.

After the separating hyperplane is determined, the points on the margins are called support vectors. The algorithm ignores the points other than those and formulates the solution as a linear combination of support vectors. Thus, SVM is unaffected by the number of features in the data. Yet, in the case of misclassified data, SVM may not find the optimal separating hyperplane.

Training SVM is achieved by solving Quadratic Programming Problem (QP) (Zanghirati and Zanni, 2003). Yet, QP is NP hard due to large computational complexity. Platt (1999) introduced Sequential Minimal Optimization (SMO) algorithm to overcome drawbacks of traditional QP. The author's method decomposes the existing QP into smaller subproblems, and solves them iteratively. Keerthi and Gilbert (2002) further improved SMO by proving the convergence of the generalized SMO. Hsu and Lin (2002) proposed decomposition method to speed up the training time of SVM.

SVM is examined in four different sections according to the categorization of James, Witten, Hastie and Tibshirani (2017). In Section 2.2.1, maximal margin classifier, in Section 2.2.2, support vector classifiers, in Section 2.2.3, support vector machines, and in Section 2.2.4, multi-class support vector machines are reviewed.

2.2.1. Maximal Margin Classifier

If we have linearly separable data with two different classes as Figure 2.3, we may find a bunch of different separating hyperplanes. Maximal margin classifier calculates the perpendicular distances of instances to the each alternative hyperplane, and select the one that maximizes the margin between training points and hyperplane.

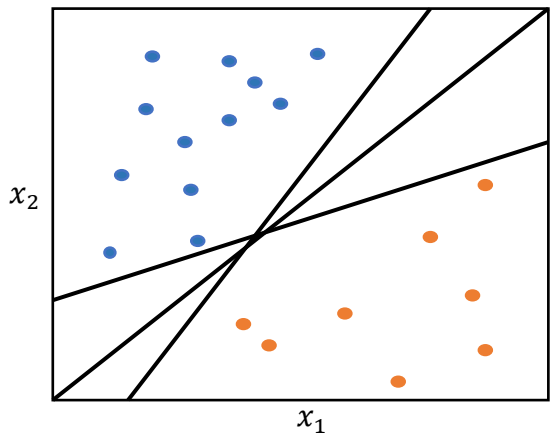

Figure 2.3. Linearly separable data

$$w.x_i + b = 0 \tag{2.2}$$

Equation (2.2) shows the formulation of hyperplane where w is the normal vector to the hyperplane, x_i is the features of the observation i, and b is bias. Each point, x_i, belongs to the one of the classes, y_i according to the (2.3) and (2.4).

$$w.x_i + b \geq 1 \text{ then } x_i \text{ belongs to class } 1, y_i = 1 \tag{2.3}$$

$$w.x_i + b \leq 1 \text{ then } x_i \text{ belongs to class } -1, y_i = -1 \tag{2.4}$$

The problem becomes a quadratic optimization problem as follows.

$$\min \quad \frac{1}{2}||w||^2 \tag{2.5}$$

s.t.

$$y_i * (w.x_i + b) \geq 1 \qquad \forall i \tag{2.6}$$

2.2.2. Support Vector Classifiers

We may have nonlinearly separable data as in Figure 2.4. In this case, maximal margin classifier may not give satisfactory results. Vapnik and Cortes (1995) proposed soft margin concept to this problem. Soft margin allows some of the data points to be classified as the wrong class at some user specified cost, C. With the introduction of soft margin, the problem in (2.5) becomes (2.7). Yet, the problem remains still quadratic optimization problem.

$$\min \quad \frac{1}{2}||w||^2 + C\sum \xi_i \tag{2.7}$$

s.t.

$$y_i * (w.x_i + b) \geq 1 - \xi_i \qquad \forall i \tag{2.8}$$

$$\xi_i \geq 0 \qquad \forall i \tag{2.9}$$

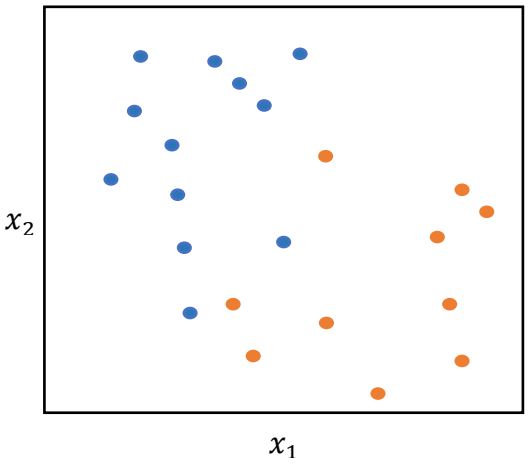

Figure 2.4. Nonlinear data

ξ_i is the slack variable which shows the location of observation i. In this situation, three possible locations exist. If $\xi_i = 0$, then the observation is classified correctly. If $1 > \xi_i > 0$, then the observation is between the hyperplane and margin. Lastly, if $\xi_i > 1$, then the observation classified wrongly. On the other hand, C represents upper limit of the total tolerance to misclassified observations. This parameter is defined by the user. Larger C values allow wider margins and lower variance. Yet, bias increases and we may face with underfitting problem. As opposed to larger C values, small C values gives narrower margins and lower bias. However, variance increases and there may be overfitting problem. Generally, cross-validation is used to find the value of C (Chollet, 2018).

Figure 2.5, shows one of the examples of possible hyperplane to linearly non-separable data.

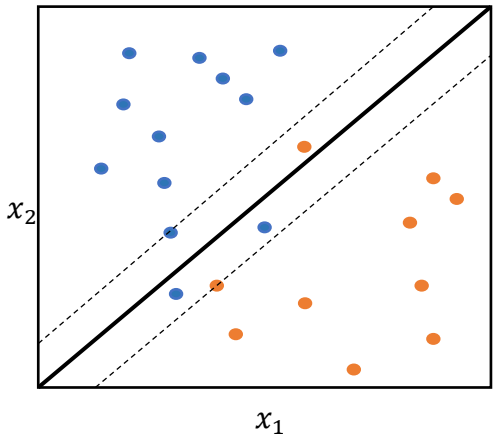

Figure 2.5. Hyperplane for linearly non-separable data

2.2.3. Support Vector Machines

A great majority of the real world problems have nonlinear class boundaries. In this situation, by assuming that linear boundary exist, the input data is mapped to a higher dimensional space through kernel trick so that separating hyperplane is linear as can be seen from Figure 2.6. After the mapping, classical SVM becomes applicable. In addition, kernel functions reduces the training time.

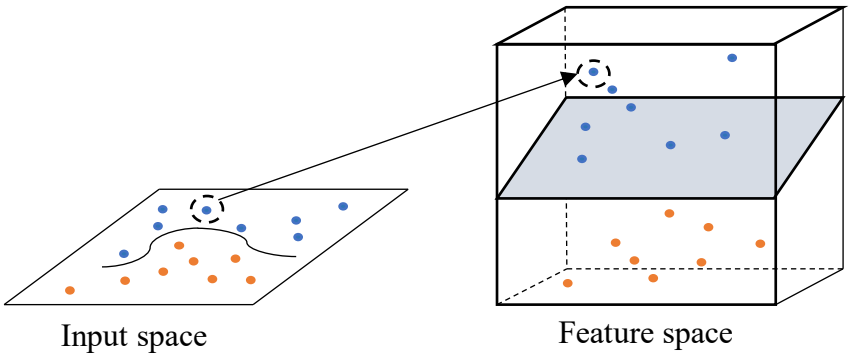

Figure 2.6. Mapping input data to feature space

There are different types of kernel functions. Yet, Géron (2018) states that polynomial and radial basis function are the most popular ones. Kavzaoglu and Colkesen (2009) and Genton (2001) give comprehensive review about the existing kernel functions.

2.2.4. Multi-Class Support Vector Machines

SVM is originally proposed as binary classification algorithm. Classifiers in Sections 2.2.1, 2.2.2, and 2.2.3 are for binary classification. In addition to binary classification, SVM is applicable for multi-class classification through decomposing the existing problem into binary problems. One versus one and one versus all classification are the most important two algorithms in this category (Milgram, Cheriet and Sabourin, 2006).

2.3. Ensembles

In ML, ensemble learning refers to the learning through combination of individual algorithms. Basic notion of this learning type is that by combining different classifiers, we tolerate the weaknesses of individual classifiers, and the overall classification performance increases. Sagi and Rokach (2018), Kulkarni and Sinha (2013) give an extensive review about this area. The important detail is that errors of the individual classifiers must be uncorrelated as correlation in errors decreases the predictive performance of the ensemble (Ali, 1995).

$$\widehat{y_i} = F(y_1, y_2, y_3 \dots y_N) \tag{2.10}$$

Ensemble learning consists of two steps, which are model training and output fusion. In the first step, classifiers to be used and training algorithm are decided. Then, in the second step, algorithm to unify the individual results is decided. Equation (2.10) shows

the mathematical representation of the whole process. y_i represents the output of the i^{th} classifier and function F represents the combination algorithm. $\hat{y}_i$ represents the combined result of the ensemble.

There are two mainstream ensemble methods. The first method combines different types of classifiers as inducers (Ladjal et al., 2016; Gama and Brazdil, 2000; Wolpert, 1992) whereas the rest combines the same type of classifiers (Li et al., 2008; Wei-Wei and Li-Na, 2012; Breiman, 1996; Freund, 1995).

2.4. Dempster-Shafer Theory (DST)

Data sources, sensors, have limited capabilities due to imperfections in both sensors themselves and environmental effects. Multi-sensor systems help to reduce such effects, provide more robust and reliable systems through the data fusion. Dempster-Shafer Theory (DST), also known as Belief Theory or Evidence Theory, is an effective fusion method to combine separate pieces of evidences coming from different sensors under uncertainty and imprecision. The theory is first introduced by Dempster (1967). Then, Shafer who was a student of Dempster, mathematically formulized the theory (Shafer, 1976). DST has a wide range of application areas such as target recognition (Chen, Cremers and Cao, 2014; Dong and Kuang, 2015), decision making (Leung, Ji and Ma, 2013; Dymova amd Sevastjanov, 2010), multi-sensor classification (Pal and Ghosh, 2001; Foucher et al., 2002), reliability analysis (Zhou et al, 2012), expert systems (Dang and Chan, 2011) and fault diagnosis (Fan and Zuo, 2006).

DST combines separate pieces of evidences coming from different sensors, and classifies them as the most likely class in the frame of discernment (FOD).

FOD is a set of all possible states of the system.

$$\Theta = \{h_1, h_2, \dots, h_M\} \tag{2.11}$$

FOD consists of M many hypotheses. These hypotheses are mutually exclusive and exhaustive. h_i represents the possible states of the system. On the other hand, power set, 2^Θ, derived from FOD contains the propositions, possible predictions, of the system. In other words, power set shows the potential result of the fusion process.

$$2^\Theta = \{\emptyset, \{h_1\}, \{h_2\}, \dots, \{h_M\}, \{h_1, h_2\}, \dots, \{h_1, h_M\}, \dots, \{h_1, h_2, \dots, h_M\}\} \quad (2.12)$$

It can be seen from (2.12) that power set is composed of 2^M many propositions. Every element of the power set is one of the subsets of FOD. To mathematically show, if $h_1 \subset \Theta$, then $h_1 \in 2^\Theta$.

Proposition H_i represents subset of the power set. Initial support degree of proposition H_i, $m(H_i)$, is defined over the power set. These support degrees are called basic probability assignments (BPA). BPA can be thought as a function that maps $m(H_i)$ as $m: 2^\Theta \rightarrow [0,1]$. BPA should satisfy the following conditions given in (2.13) and (2.14).

$$m(\emptyset) = 0 \quad (2.13)$$

$$\sum_{H_i \subseteq 2^\Theta} m(H_i) = 1 \quad (2.14)$$

Condition (2.13) satisfies the nonnegativity, and condition (2.14) ensures the unity property.

Any BPA that satisfies $m(H_i) > 0$ is called focal element.

Getting BPAs from different sensors and combining them, belief and plausibility of each hypothesis are found. Belief of proposition H_i, $Bel(H_i)$, shows the total belief assigned to proposition H_i. On the other hand, plausibility of H_i, $Pl(H_i)$, shows the upper probability that proposition H_i cannot be regarded as true anymore. They are

calculated according to (2.15) and (2.16). In addition, conditions (2.17) and (2.18) hold true for both of them.

$$Bel(H_i) = \sum_{H_l \subset H_i} m(H_l) \tag{2.15}$$

$$Pl(H_i) = \sum_{H_l \cap H_i \neq \emptyset} m(H_l) \tag{2.16}$$

$$Bel(H_i) \leq Pl(H_i) \tag{2.17}$$

$$Pl(H_i) = 1 - Bel(\overline{H_i}) \tag{2.18}$$

Uncertainty in DST is represented by uncertainty interval. The relationship between $Bel(H_i)$ and $Pl(H_i)$ can be seen from Figure 2.7. $Bel(H_i)$ and $Pl(H_i)$ constitute lower and upper bounds of uncertainty interval, respectively.

$$\mu(H_i) = Pl(H_i) - Bel(H_i) \tag{2.19}$$

$\mu(H_i)$ represents the uncertainty in the evidence.

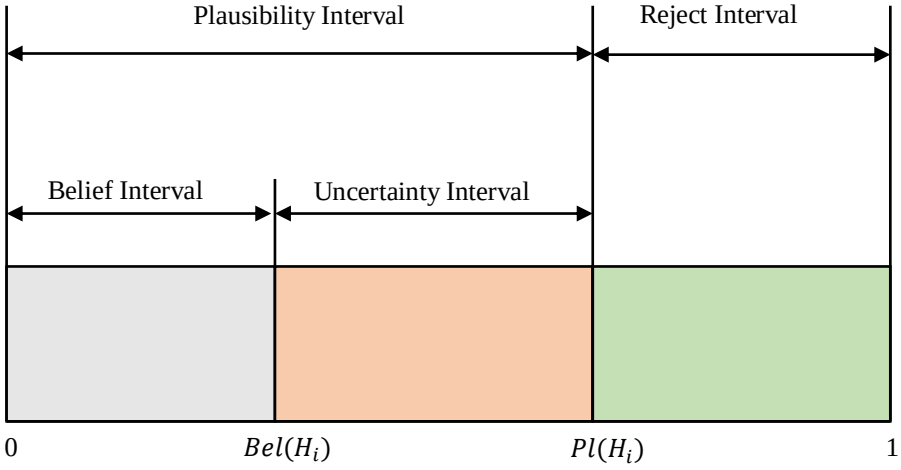

Figure 2.7. Relationship between belief and plausibility (Chen et al., 2017)

To make interpretation of $Bel(H_i)$ and $Pl(H_i)$ more clear, we may give an example. Suppose that we have a hypothesis that states as follows:

H_1: The target is a type of Bomber Aircraft

$[Bel(H_1), Pl(H_1)] = [0.6,0.8]$

For this example, we have an evidence that states that the target is a type of Bomber Aircraft with probability of 0.6. However, the type of the target is different than the Bomber Aircraft with probability of 0.2, $(1 - Pl(H_1))$. Also, we are uncertain our hypothesis with probability of 0.2, $(Pl(H_1) - Bel(H_1))$.

From the example, we can deduce the followings:

$Bel(H_i)$: Represents the supporting evidence

$(1 - Pl(H_i))$: Represents the contrary evidence

As the uncertainty interval gets smaller, we are more certain about our belief. On the other hand, as the uncertainty interval gets larger, we are more uncertain about our belief.

To fuse the evidences coming from different sensors, DST combination rule is proposed. Assume that there are two sensors and we have a FOD as $\Theta = \{H_1, H_2, \dots, H_M\}$. DST combination rule suggests condition (2.20) to fuse the evidences. Note that DST explicitly assumes that all sensors are independent. This is an important assumption of DST. DST combination rule is also referred as *Orthogonal Sum of Evidences*.

$$m(H) = \begin{cases} \frac{1}{1-k} * \sum_{H_i \cap H_j = H} m_1(H_i) * m_2(H_j) & if\ H \neq \emptyset \\ 0 & if\ H = \emptyset \end{cases} \quad (2.20)$$

Here, k is a measure of conflict between evidences and $\frac{1}{1-k}$ is the normalization factor, which ensures the unity property stated in condition (2.14). It is calculated according to (2.21).

$$k = \sum_{H_i \cap H_j = \emptyset} m_1(H_i) * m_2(H_j) \quad (2.21)$$

DST combination rule satisfies both commutative and associate law.

Equations (2.20) and (2.21) shows the combination rule and calculation of the conflicting degree for two sources of information, respectively. To extend the theory to multi-sensor, one need to combine the evidences iteratively over all sensors.

After getting belief and plausibility of all of the propositions, DST applies some decision making rule to get the final classification result of the system. Here, there are two common decision making rules, which are choosing the proposition with the maximum belief value or maximum plausibility value.

Unlike probabilistic data fusion, DST does not need prior probabilities. The algorithm recognizes that each evidences may have different levels of detail and assigns probabilities to pieces of evidence only if there is supporting information. It can efficiently deal with the imprecise and uncertain information (Khaleghi et al., 2013). However, due to complex monitoring environment and limited accuracy of the sensors, DST may give counterintuitive results. The first critic about DST belongs to Zadeh (1979). He shows that when the evidences are conflicting, DST gives insufficient results (Zadeh, 1979; Zadeh, 1984; Zadeh, 1986). After his critics, many researches focused on this topic. Common paradoxes in DST are categorized into four groups (Li et al., 2016).

Complete Conflict Paradox

When conflicting degree is equal to 1, DST combination rule cannot be applied. This paradox is called as “Complete Conflict Paradox”.

0 Trust Paradox

This paradox is also called as “One Bullet Veto”. When one of the evidences are totally denied by one of the sensors, the resulting fused mass of it always gets 0.

1 Trust Paradox

Under highly conflicting evidences, DST may give total mass to one of the evidences even if they are poorly justified by the all sensors. This situation is called “1 Trust Paradox”.

High Conflict Paradox

Even if the majority of the sensors justify one of the evidences, DST may give a small combined mass to it due to high conflict degree. This paradox is called “High Conflict Paradox”.

Existing literature is divided into two about the main causes of the conflict as can be seen in Figure 2.8. Some of the researches state that conflicting situation is the result of the imprecision in sensors. This type of researches focus on new modified versions of DST based on evidence correction before combination. On the other hand, the second group of researches states that the main problem comes from normalization step of DST and investigate new conflict redistribution strategies. Apart from these, some of the researches take into account both complex monitoring environment and imprecision of sensors, recently. In Section 2.4.1, we review the literature on evidence correction. In Section 2.4.2, alternative conflict redistribution strategies are overviewed. In Section 2.4.3, studies that take into account both evidence correction and conflict redistribution are reviewed.

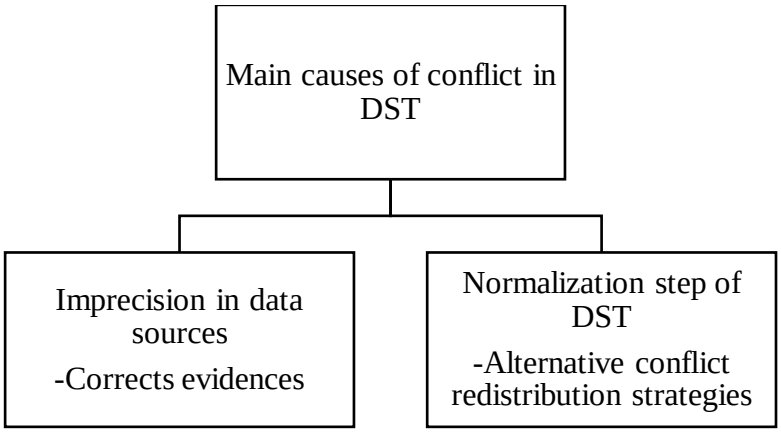

Figure 2.8. Main causes of conflict in DST

2.4.1. Modified DST Based on Evidence Correction

Researches that fall into this category regard DST combination rule as a good enough method to combine evidences. Yet, they argue that insufficient result comes from the imprecision in sensors. Studies in this category mainly deal with the evidence correction before the combination rule is applied. In this section, selected works in the literature will be presented.

Mathematical theory of evidence: To deal with the imprecision in sensors, evidence discounting before the combination was first proposed by Shafer (1976).

$$m'(A) = t * m(A) \quad \forall A \subset FOD \tag{2.22}$$

$$m'(FOD) = t * m(FOD) + (1 - t) \tag{2.23}$$

Where t is the reliability of the sources where $0 \leq t \leq 1$.

Shafer's approach discounts the evidences by multiplying the original evidence with the reliability of the source, and devotes the remaining discounted mass, $1 - t$, to

FOD. When $t = 0$, that is the source is totally unreliable, the algorithm gives the total mass to FOD. In this situation, the mass function becomes vacuous, $m(FOD) = 1$, and total ignorance situation occurs. When vacuous belief assignment occurs, basic probability assignment gives no information. On the other hand, when $t = 1$, that is the source is totally reliable, the original masses remain the same.

- ✓ This algorithm increases the global ignorance when $m(FOD) < 1$ and $t < 1$ due to the fact that $t * m(FOD) + (1 - t) > t * m(FOD) + (1 - t) * m(FOD) = m(FOD)$.
- ✓ Reliability of the sources are taken into account.
- ✓ Assessing the reliability of the sources is hard and subjective.
- ✓ Vacuous belief assignment occurs when the source is totally unreliable.

Combining belief functions when evidence conflicts: Murphy (2000) proposed an evidence correction algorithm based on both simple averaging of the BPAs and classical DST. In his approach, firstly, the averages of the BPAs are calculated, and then these averaged BPAs are combined with themselves by classical DST combination rule.

$$m(A) = \frac{1}{k}\sum_{i=1}^{k} m_i(A) \tag{2.24}$$

Where k is the number of sources.

- ✓ The algorithm shows mass convergence property. That is the combined masses can be greater than that the original masses.
- ✓ Algorithm can deal with 0 Trust Paradox.
- ✓ All evidences are regarded as equally important, which is not reasonable.
- ✓ Reliability of the sources are not taken into account.

Combining belief functions based on evidence distance: Yong et al. (2004) criticize Murphy's (2000) work for treating all sources of evidences equally by stating that each evidence may have different importance. Their approach was inspired from Voorbraak (1991) in the sense that similarity of the evidences are assigned over distance functions.

Let $SIM(m_i, m_j)$ be similarity degree between evidences.

$$SIM(m_i, m_j) = 1 - d(m_i, m_j) \tag{2.25}$$

Where $d(m_i, m_j)$ is the distance between evidences.

To calculate the distances, Jousselme Distance is used.

Having defined the similarity degrees, similarity matrix, SMM, is derived.

$$SMM = \begin{pmatrix} 1 & \cdots & SIM(m_1, m_k) \\ \vdots & \ddots & \vdots \\ SIM(m_k, m_1) & \cdots & 1 \end{pmatrix} \tag{2.26}$$

By the help of the similarity degrees, support and credibility degrees of each evidence are calculated.

$$\sup(m_i) = \sum_{j=1, j\neq i}^{k} \mathrm{SIM}(m_i, m_J) \tag{2.27}$$

$$crd_i = \frac{\sup(m_i)}{\sum_{j=1}^{k} \sup(m_j)} \tag{2.28}$$

Credibility degree is just the normalized version of support degree of evidence m_i. It represents the similarity between m_i and all of the other evidences.

Finally, they discount the evidences according to (2.29).

$$m'(H_l) = \sum_{i=1}^{k} crd_i * m_i(H_l) \tag{2.29}$$

After getting the discounted masses, they combine the evidences according to the classical DST.

- ✓ Reliability of the sources are assigned through evidence distances.
- ✓ The effect of conflicting pieces of evidences coming from different sources are decreased at combination stage.
- ✓ The commutative and associative properties are not preserved. That is the ordering of the evidences may change the result.

Refined modelling of sensor reliability in the belief function framework using contextual discounting: Mercier et al. (2008) introduced contextual discounting which is more comprehensive version of the classical discounting operation. This type of discounting recognizes that each sources of information may have different reliability about each type of target. For example, some of the sensors may have higher reliability for recognizing if the target is airplane and lower reliability for recognizing if the target is rocket.

Reliability degrees of sources, which represent the reliabilities based on different contexts, are held in a vector. Once the reliability vector is built, evidences are discounted.

- ✓ Reliability of the sources are taken into account.
- ✓ Contrary to classical DST, Mercier et al. (2008) assign reliability for each element of FOD.
- ✓ It is hard to obtain reliability vector in real life.

Decision rule for pattern classification by integrating interval feature values: Horiuchi (1998) proposed an integrated method that modifies evidences according to the source value by stating that source value of each evidence are not equal.

$$q(H_l) = \sum_{H_i \cap H_j = H_l} w_i * m_i(H_i) + w_j * m_j(H_j) \tag{2.30}$$

Where w_i is the weight of the source value.

Horiuchi (1998) states that there are many decision rules to assign the source value. For example, he proposes (2.31) to calculate the source values.

$$w_i = \frac{\frac{1}{{\sigma_i}^2}}{\sum_j \frac{1}{{\sigma_j}^2}} \tag{2.31}$$

Where ${\sigma_i}^2$ is the variance.

- ✓ Reliability of the sources are taken into account through the source values.
- ✓ There are many alternatives to determine the source values. Yet, it is uncertain that which of them is more appropriate.
- ✓ Commutative property is preserved whereas associative property is not.

Evidence reasoning for temporal uncertain information based on relative reliability evaluation: Fan et al. (2018) state that the reliabilities of the sources of information may change with the time, and while performing fusing operations, the algorithm should be adaptive to dynamic outputs of the sensors. Thus, they proposed a new reliability evaluation and evidence discounting approach that can deal with the sensor outputs dynamically. First, evidence reliabilities are calculated based on a new intuitionistic fuzzy multiple criteria decision making (IFMCDM) model. Once the

reliabilities are calculated, they are discounted according to the Shafer's (1976) discounting method. Finally, discounted evidences are combined according to the classical DST combination rule.

- ✓ Dynamic reliabilities of the sources are taken into account.
- ✓ The proposed algorithm is computationally complex.

An improved Dempster–Shafer approach to construction safety risk perception: Zhang et al. (2017) point out that classical DST may give good results with low conflicting evidences. By considering this, they propose a hybrid algorithm shown in Figure 2.9.

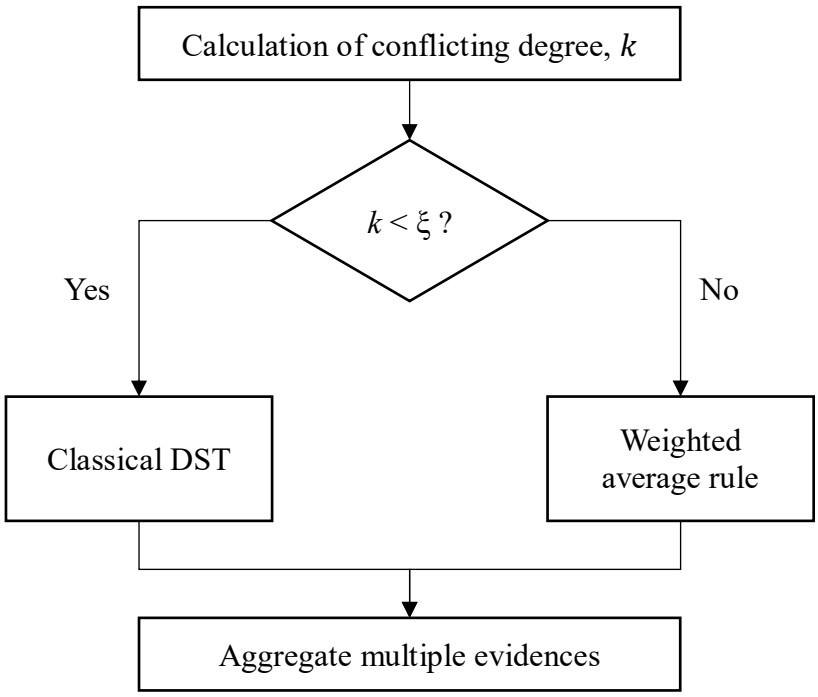

Figure 2.9. Flowchart of the proposed algorithm by Zhang et al. (2017)

According to their algorithm, if the conflicting degree, k, is below a predefined threshold, then classical DST is activated. Otherwise, weighted average rule based on distances is employed. Threshold value, ξ, determines which algorithm to apply.

- ✓ Reliabilities of the sources of information are taken into account by evidence distances.
- ✓ There is no certain way of determining the threshold value, ξ.

Weighted evidence combination rule based on evidence distance and uncertainty measure: An application in fault diagnosis: Chen et al. (2018) proposed a two-step modification algorithm. In the first step, credibility degrees are calculated based on evidence distance. After that, uncertainty in the evidences are added to the difference between plausibility and belief of evidences. Their work is inspired from Yong et al. (2004). Contrary to Yong et al. (2004), they proposed to discount the evidences according to the uncertainty of each BPAs.

- ✓ Reliability of the sources are assigned through evidence distances.
- ✓ The commutative and associative properties are not preserved. That is the ordering of the evidences may change the result.

Table 2.2 shows the summary of the literature on modified DST based on evidence correction.

Table 2.2. *Review of modified DST based on evidence correction*

Paper	**Reliability of Sensors**	**Proposed Way for Assigning Reliabilities**	**Proposed Way for Evidence Correction other than Reliability of Sensors**
Shafer (1976)	✓	×	×
Murphy (2000)	×	×	Simple averaging the evidences
Yong, WenKang, ZhenFu and Qi (2004)	Credibility Degrees	Jousselme Distance	×
Mercier, Quost and Denoeux (2006)	Contextual Discounting	×	×
Horiuchi (1998)	Source Value	Source Variances	×
Fan, Song, Lei, Wang and Bai (2018)	Dynamic Reliability	Intuitionistic Fuzzy MCDM	×
Zhang, Ding, Wu and Miroslaw (2017)	Evidence Distance	Euclidean Distance	Hybrid Model
Chen, Diao and Sang (2018)	Credibility Degrees	Jousselme Distance	Uncertainity is added to the credibility degrees

2.4.2. Modified DST Based on Conflict Redistribution

On the contrary to the first part, some of the researches predicate insufficient result of DST on normalization step. Classical DST combination rule normalizes partial conflicts in proportion to total conflict. In some cases, this approach causes unreasonable results. To overcome this problem, many researches proposed alternative combination rules. In this section, selected works in the literature will be presented.

The transferable belief model: In 1994, Smets and Kennes proposed an alternative combination rule to the classical DST combination rule. Their combination rule is actually the non-normalized version of the Dempster's rule. Yet, they assume that FOD is an incomplete set, and devote total conflict to unknown propositions. That is conflicting mass is assigned to empty set.

$$m(\emptyset) \equiv k_{12} = \sum_{\substack{H_i, H_j \in 2^{\Theta} \\ H_i \cap H_j = \emptyset}} m_1(H_i) * m_2(H_j) \tag{2.32}$$

$$m(H_l) = \sum_{\substack{H_i, H_j \in 2^{\Theta} \\ H_i \cap H_j = H_l}} m_1(H_i) * m_2(H_j) \tag{2.33}$$

After getting the combined masses, they also proposed decision-making rule based on pignistic probabilities calculated as follows:

$$BetP(H_l) = \sum_{H_i \in 2^{\Theta}} m(H_i) * \frac{|H_l \cap H_i|}{|H_i|} \tag{2.34}$$

Where $|H_i|$ is the cardinality set of H_i.

- ✓ Although being efficient in overcoming paradoxes, the proposed method increases the uncertainty due to unknown propositions.

- ✓ Reliability of the sources are not taken into account.
- ✓ The commutative and associative properties are preserved.
- ✓ Open world assumption is made.
- ✓ Conflict is distributed globally.
- ✓ Normalization factor is disregarded.

On the Dempster-Shafer framework and new combination rules: Yager (1987) stated that conflicting situation is not reliable, and it should be added to the total ignorance, or universal set, as discounting term. Unlike Smets and Kennes (1994), he does not make open world assumption and gives zero mass to empty set.

$$m(\emptyset) = 0 \tag{2.35}$$

$$m(H_l) = \sum_{\substack{H_i, H_j \in 2^{\Theta} \\ H_i \cap H_j = H_l}} m_1(H_i) * m_2(H_j) \tag{2.36}$$

$$m(\Theta) = m_1(\Theta) * m_2(\Theta) + \sum_{\substack{H_i, H_j \in 2^{\Theta} \\ H_i \cap H_j = \emptyset}} m_1(H_i) * m_2(H_j) \tag{2.37}$$

- ✓ Reliability of the sources are not taken into account.
- ✓ Open world assumption is not made.
- ✓ Conflict is distributed globally.
- ✓ Normalization factor is disregarded.
- ✓ The algorithm is intolerant to errors in the sources of information. Thus, the efficiency of the algorithm drops as the number of sources increase.

Representation and combination of uncertainty with belief functions and possibility measures: Dubois and Prade (1988) states if two of the sources conflict with each other, one of them is not reliable. Their formulation consists of both

conjunctive and disjunctive combination of the masses where conjunctive combination of the masses assumes that both of the propositions are true, and disjunctive combination assumes that at least one of the propositions is true.

They distribute the conflicting mass to the focal elements of disjunctive combination of the masses.

$$m(\emptyset) = 0 \tag{2.38}$$

$$m(H_l) = \sum_{\substack{H_i, H_j \in 2^\Theta \\ H_i \cap H_j = H_l \\ H_i \cap H_j \neq \emptyset}} m_1(H_i) * m_2(H_j) + \sum_{\substack{H_i, H_j \in 2^\Theta \\ H_i \cup H_j = H_l \\ H_i \cap H_j = \emptyset}} m_1(H_i) * m_2(H_j) \tag{2.39}$$

- ✓ Conflict is assigned partially.
- ✓ Reliability of the sources is not taken into account.
- ✓ Commutative property is preserved whereas associative property is not.
- ✓ Open world assumption is not made.
- ✓ The method provides tradeoff between precision and reliability by the way of distributing the conflict.

Interdependence between safety-control policy and multiple-sensor schemes via Dempster-Shafer theory: Inagaki (1991) proposed a general formulation for combination rules and distribution of conflict. His formula distributes the mass of empty set, or conflict, globally after the conjunctive combination of masses.

$$m(\emptyset) = 0 \tag{2.40}$$

$$m(H_l) = \sum_{\substack{H_i, H_j \in 2^\Theta \\ H_i \cap H_j = H_l}} m_1(H_i) * m_2(H_j) + k * w(H_l) \tag{2.41}$$

Where $\sum_{H_l \subseteq 2^\Theta} w(H_l) = 1$ and $w(H_l) \geq 0$.s

- ✓ Reliability of the sources are taken into account.
- ✓ There is no consensus about choosing the weights to distribute the conflict.
- ✓ Open world assumption is not made.
- ✓ Commutative property is preserved whereas associative property is not.
- ✓ Conflict is assigned proportionally.

Proportional conflict redistribution rules for information fusion: In addition to Inagaki (1991)'s general formulation, Smarandache and Dezert (2006) proposed another general formulation. Their formula distributes the conflict to involved focal elements. Yet, Leung, Ji and Ma (2013) criticize the way of distributing the conflict by saying that the proposed general formulations should distribute the conflict all over the sets.

$$m(\emptyset) = 0 \tag{2.42}$$

$$m(H_l) = \sum_{\substack{H_i, H_j \in 2^\Theta \\ H_i \cap H_j = H_l}} m_1(H_i) * m_2(H_j) + \sum_{\substack{H_i, H_j \in 2^\Theta \\ H_i \cap H_j = \emptyset}} \left[\frac{m_1(H_i)^2 * m_2(H_j)}{m_1(H_i) + m_2(H_j)} + \frac{m_1(H_j) * m_2(H_i)^2}{m_1(H_j) + m_2(H_i)}\right] \tag{2.43}$$

- ✓ Conflict is assigned proportionally.
- ✓ Open world assumption is not made.
- ✓ Commutative and associative properties are preserved.
- ✓ Reliability of the sources are not taken into account.

Strategies for combining conflicting dogmatic beliefs: Like Smarandache and Dezert (2006), Josang, Daniel and Vannoorenberghe (2003) are inspired by Inagaki (1991). Their algorithm is the same as the Inagaki (1991)'s except for the way of calculating the weights. The distributed conflict is as follows:

$$m(\emptyset) = 0 \tag{2.44}$$

$$m(H_l) = \sum_{\substack{H_i,H_j \in 2^\Theta \\ H_i \cap H_j = H_l \\ H_i \cap H_j \neq \emptyset}} m_1(H_i) * m_2(H_j) + w(H_l) * \sum_{\substack{H_i,H_j \in 2^\Theta \\ H_i \cap H_j = \emptyset}} m_1(H_i) * m_2(H_j) \tag{2.45}$$

Where $w(H_l) = \frac{1}{2} * (m_1(H_l) + m_2(H_l))$.

- ✓ Conflict is globally proportionally.
- ✓ Reliability of the sources are not taken into account.
- ✓ Commutative and associative properties are preserved.
- ✓ Open world assumption is not made.

Robust combination rules for evidence theory: Florea et al. (2009) proposed a class robust combination rule that distributes the conflict to all over the set. In their formulation, they use the conflict between evidences as the weight.

$$m(\emptyset) = 0 \tag{2.46}$$

$$m(H_l) = \frac{1}{1-k-k^2} * \sum_{\substack{H_i,H_j \in 2^\Theta \\ H_i \cap H_j = H_l}} m_1(H_i) * m_2(H_j) + \frac{k}{1-k-k^2} * \sum_{\substack{H_i,H_j \in 2^\Theta \\ H_i \cup H_j = H_l}} m_1(H_i) * m_2(H_j) \tag{2.47}$$

Conflicting factor, k, is calculated according to the classical DST formulation. The proposed formula behaves like disjunctive as the conflicting degree increases. On the

other hand, as the conflicting degree gets lower and evidences become more reliable, the combination rule behaves like a conjunctive.

- ✓ Commutative property is preserved whereas associative property is not.
- ✓ Reliability of the sources are taken into account through the weights, which are basically the function of conflict.
- ✓ Open world assumption is not made.
- ✓ Conflict is assigned globally.

2.4.3. Modified DST Based on both Evidence Correction and Conflict Redistribution

The researches that fall into this category believe that adapting only one of evidence correction and conflict redistribution strategy does not deal with insufficiency of DST fully. Thus, they address both aspects. In this section, selected works are presented.

The improvement of DS evidence theory and its application in IR/MMW target recognition: Li et al. (2016) provide a new conflict redistribution and decision making strategy by taking into account sensor priorities and evidence credibility. In their approach, sensor priorities are assigned according to the type and precision of the sensors. The proposed algorithm modifies the weights before combination based on their consistency and reliability indexes calculated according to the algorithm proposed by Yong et al. (2004). Euclidean distance function is used to evaluate the distances between evidences. In combination stage, conflict is distributed globally. However, when the number of sensors increases, the method becomes insufficient.

Conflicting information fusion based on an improved DS combination method: Chen et al. (2017) revise two pieces of evidences separately by weighted Minkowski and Betting Commitment distance functions. Then, they combine these two by

modified combination rule that assigns conflict locally. Their algorithm works well under common paradoxes.

Combination of classifiers with optimal weight based on evidential reasoning: Optimal Weighted Dempster-Shafer (OWDS) is developed by Liu et al. (2018). This method optimizes the classifier weights by minimizing the distance between combination result and the true label of the data. Distance is calculated according to Jousselme's distance function. However, the proposed method has computational complexity due to the optimization procedure and cannot be applicable to larger datasets.

Data Fusion Method Based on Improved D-S Evidence Theory: Zhang et al. (2018) modify evidences according to Bhattacharyya distance and combines through new combination rule. However, the proposed algorithm gives reasonable results with small datasets.

A robust DS combination method based on evidence correction and conflict redistribution: Ye et al. (2018) modify evidences according to reliability index and average mass assignment. To calculate reliability index, Matusita distance is used and difference between evidences is calculated. Also, to consider the different consistency degrees of evidences, average mass assignment is calculated. After, they modify the evidences according to reliability index and average mass assignment. Finally, weighted mass assignment is employed to combine the modified evidences. Their method is proven to be efficient.

CHAPTER 3

MULTI-SENSOR TARGET CLASSIFICATION MODELS

In this chapter, we introduce two multi-sensor target classification algorithms. Both algorithms classify each sensors' dataset by ensemble, and then combine them with modified DST. They differ from each other in the modification of the evidences in DST. The first algorithm modifies the evidences by credibility degrees of the sensors whereas the second one modifies by both credibility degrees and accuracies of the sensors.

3.1. Ensemble of Classifiers with Modified Distance Function (ECMDF): Overview

Ensemble of Classifiers with Modified Distance Function (ECMDF) is the first proposed multi-sensor target classification approach. The proposed algorithm is ML based. Thus, we will explain it in two parts which are training and test phases.

Training phase of ECMDF:

Training phase of ECMDF consists of three main steps as shown in Figure 3.1. The algorithm starts with reading the multi-sensor datasets in Step 0. In Step 1, sensor datasets are split as training and test. Based on the training datasets, both ANN and SVM are trained separately for each sensor. After training both of the classifiers for each sensor, the one with the higher classification accuracy based on training datasets is chosen for each sensor. With the chosen classifier, each sensors' accuracy ratios, probabilistic classification results and predicted classes are determined based on training datasets by tuned classifier. In Step 2, training outputs of all sensors are

combined through modified DST. In this step, firstly pairwise correlation coefficients between sensors' predicted classes are calculated over all training datasets. Then, correlation matrix is built. The resulting matrix's elements comprise of values between -1 and 1. They are then scaled between user specified intervals. Scaled correlation coefficient matrix's elements are used as p parameter of L_p metric while calculating the pairwise distances between evidences of sensors for each target. Based on distances, pairwise distance matrixes are built for each target. Using the distance matrixes, average credibility degrees of all sensors are calculated for all targets. By using the calculated average credibility degrees, probabilistic classification results of sensors are discounted, and combined with the classical DST combination rule. At this stage, the proposed algorithm moves to the beginning of Step 2, and Step 2 is repeated for different values of scaling intervals of correlation coefficients. After Step 2 is repeated for all scaling intervals, the one that maximizes the number of correct predictions is chosen, which ends up the training phase of ECMDF.

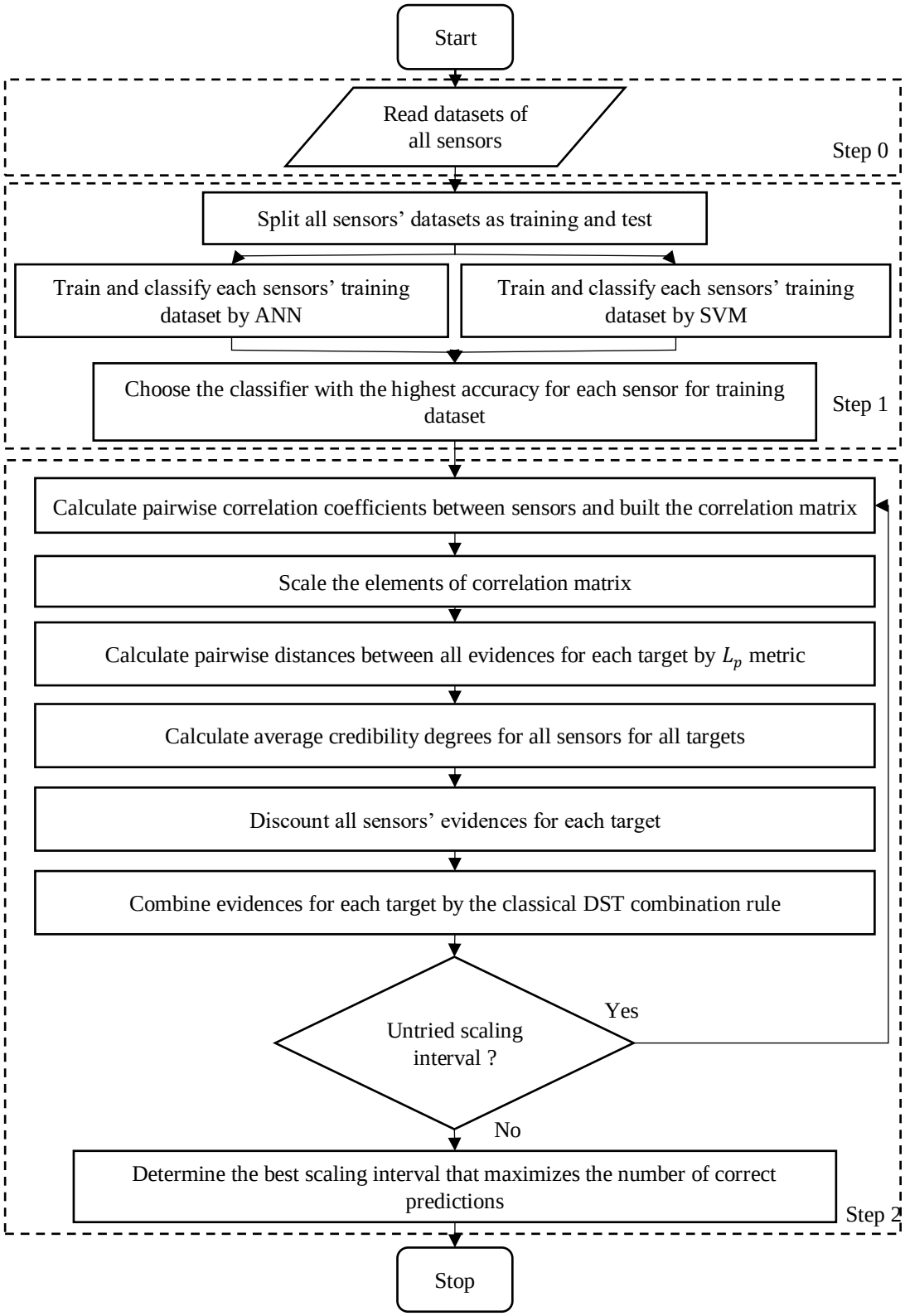

Figure 3.1. Flowchart of the training phase of ECMDF

Test phase of ECMDF:

In the training phase, ECMDF identifies the classifier for each sensor, ANN or SVM, and determines sensor accuracies and credibility degrees to be later used in the test phase to discount the evidences. The flowchart of test phase of ECMDF can be seen in Figure 3.2. Note that before the test phase, the training phase must be completed.

The test phase starts with Step 3 by probabilistically classifying each sensor test dataset by the classifier determined in the training phase. In this step, all evidences from multiple sensors are derived for each target. If there are i sensors, then at the end of this step, there will be i different evidences for the same target. To find the final classification result, we need to combine i many number of evidences for each target. After all evidences are discounted with calculated credibility degrees in training phase for each target, they are combined by the classical DST combination rule. After performing the combination, ECMDF gives the final combined evidence of each target. Finally, the proposition with the maximum belief value in combined evidence is chosen for each target.

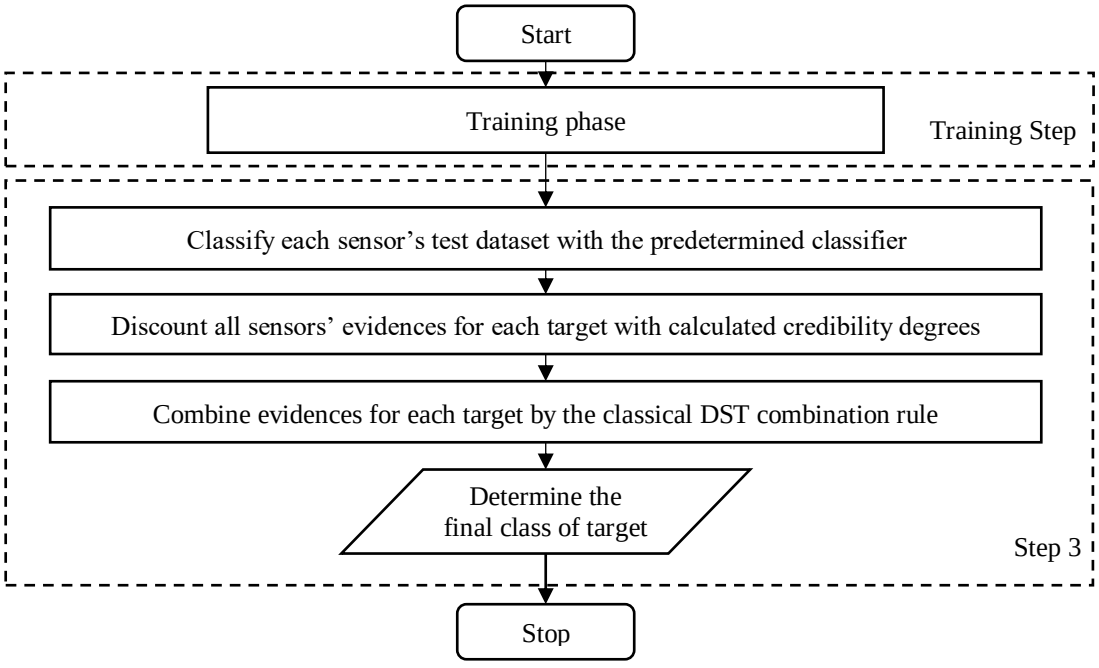

Figure 3.2. Flowchart of the test phase of ECMDF

3.2. Relationship between Correlation Coefficients and Evidence Distances

The classical DST may give counterintuitive results due to complex monitoring environment and limited accuracy of the sensors in paradoxical and high conflicting situations. Thus, in order to get satisfactory results when the evidences conflict, classical DST must be modified. In the literature, there are many studies on evidence correction, conflict redistribution or both to overcome this situation.

If we consider all of those studies generally, we can see that most of them propose discounting evidences through reliability degrees of information sources to handle imprecision in sensors. For example, Yong et al. (2004), Zhang et al. (2017) and Chen et al. (2018) use evidence distances to assign reliability degrees. Fan et al. (2018) employ IFMCDM. Horiuchi (1998) propose calculating the source values, i.e. source weights, through source variance in the readings.

Yong et al. (2004) use credibility degrees to represent the reliability degrees of the sensors. By inspiring from Yong et al.'s (2004) work, we propose a new way of calculating credibility degrees to discount the evidences. In our approach, we employ L_p metric, in which p parameter learns its value from the training datasets of sensors through the correlation coefficients between sensors. In the distance function we propose, correlation between sensors are calculated firstly. Then, they are scaled between user specified intervals.

The logic behind the training of the p parameter is depicted in Figure 3.3, in which relationship between correlation coefficients and consistency between sensors are presented. For example, if the correlation between two sensors are high, consistency between them is high, too. Therefore, distance between evidences coming from these sensors should be small. On the other hand, if the correlation coefficients between sensors are low, then the distance between evidences coming from them should be higher.

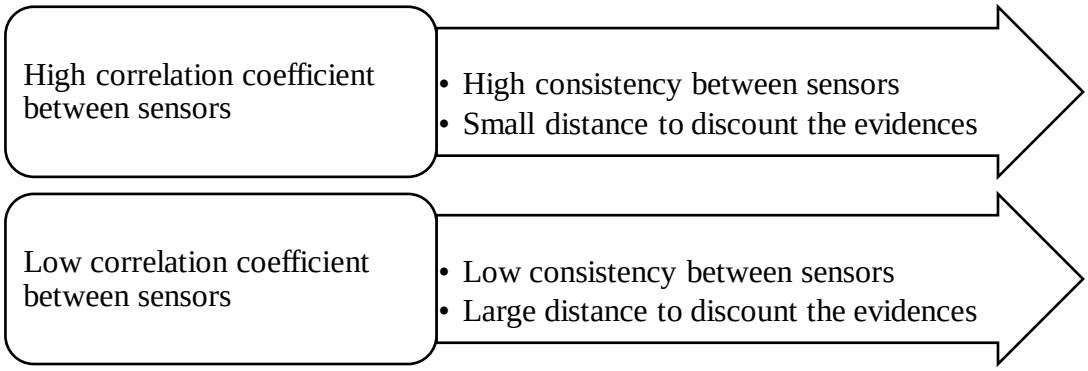

Figure 3.3. Relationship between correlation coefficients and evidence discounting

Given sample vectors A and B, Figure 3.4 shows how the distance between vectors change with varying p parameter of L_p distance metric. As p increases, the distance between the vectors decreases. Notice that also as correlation between sensors increase, numerical value of correlation coefficient increases. In order to apply the logic explained above, correlation coefficients between sensors is used to determine the alternative p values of L_p metric.

$A = [14, 22, 35]$

$B = [5, 2, 9]$

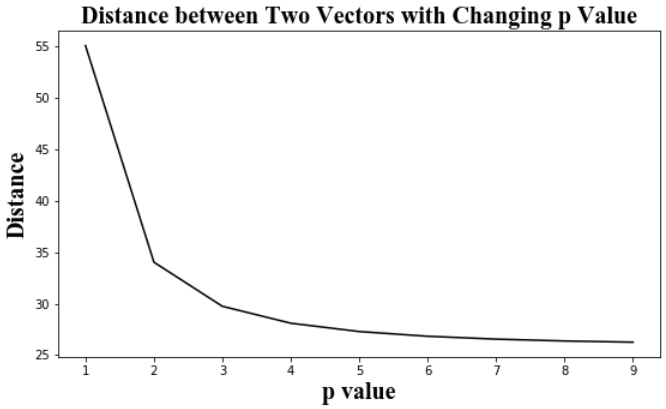

Figure 3.4. Change in distance with increasing p parameter

3.3. Assumptions

We made the following assumptions for our models.

- ✓ Sensors are heterogeneous, and they provide different types information about the same potential target. That is each sensor provide different feature information.
- ✓ Sensors are independent.
- ✓ FOD of sensors are same.
- ✓ Datasets of sensors are equal long.
- ✓ Sensors provide information for only numerical features.
- ✓ Targets belong to only one of the classes. They do not have multiple labels, or classes.
- ✓ There is not open world assumption. That is all possible classes for targets are contained in FOD.

3.4. Notations Used

The following notations are used for ECMDF algorithm:

$nbSensors$	Number of sensors in multi-sensor system
$nbTrainingTargets$	Total number of targets in training dataset of each sensor
$nbTestTargets$	Total number of targets in test dataset of each sensor
$nbPropositions$	Number of propositions that represents total number of potential results/classes of the fusion process
$sInterval$	List that contains different intervals of scaling operation
i, j	Indices for sensors, $i, j = 1, \dots, nbSensors$
k, l	Indices for targets in the training dataset, $k, l = 1, \dots, nbTrainingTargets$
m, n	Indices for targets in the test dataset, $m, n = 1, \dots, nbTestEvidences$

p, r	Indices for propositions, $p, r = 1, \dots, nbPropositions$
$u = (x, y)$	Represents the scaling interval whereas x and y show the lower and the upper bounds for scaling, respectively.
t	Index for the number of combinations.
H_p	Class of proposition p.
m_{kip}	BPA of sensor i for proposition p and target k for training dataset.
m_{ki}	Piece of evidence of sensor i for target k for training dataset.
mC_{ki}	Predicted class of target k based on m_{ki} by sensor i.
acc_i	Accuracy of sensor i based on training dataset.
$CORR$	Pairwise correlation matrix between all sensors.
$corr_{ij}$	Correlation coefficient between sensor i and j's predicted classes for all training evidences.
$sCorr_{ij}$	Scaled correlation coefficient between sensor i and j's predicted classes for all training evidences.
d_{kij}	Distance between evidences of sensor i and j for target k.
DIS_k	Pairwise distance matrix between all sensors' evidences for target k.
s_{kij}	Similarity degree between evidences of sensor i and j for target k.
SIM_k	Pairwise similarity matrix between all sensors' evidences for target k.
sup_{ki}	Support degree of sensor i's evidence for target k.
crd_{ki}	Credibility degree of sensor i's evidence for target k.
crd_i	Average credibility degree of sensor i for all of targets.
m'_{kp}	Discounted BPA for target k for proposition p for training datasets of all sensors.

k_{kt}	Combined conflicting degree of all sensors for target k in combination step t.
m''_{kpt}	For target k, BPA of proposition p in t^{th} combination for training datasets of all sensors.
m''_{kp}	For target k, BPA of proposition p in final combination for training datasets of all sensors.
$fCrd_i$	Calculated credibility degree of sensor i after training.
mt_{mip}	For target m, sensor i's BPA for proposition p for test dataset.
mt_{mi}	Piece of evidence of sensor i for target m for test dataset.
mt'_{mp}	Discounted BPA for target m for proposition p for test datasets of all sensors.
mt''_{mpt}	For target m, BPA of proposition p in t^{th} combination for test datasets of all sensors.
mt''_{mp}	For target m, BPA of proposition p in final combination for test datasets of all sensors.
$aver_{mp}$	Average BPA for target m for proposition p.
$Bel(H_p)_m$	Belief degree of proposition p for target m.

3.5. Objective Function

The objective of the proposed algorithm is to maximize the number of correctly predicted targets.

$$y_m = \begin{cases} 1 & \text{if class of target } m \text{ is correctly predicted} \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

We define the objective function of ECMDF as follows:

$$\max \sum_{m=1}^{nbTestTargets} y_m \tag{3.2}$$

3.6. Details of ECMDF

The detailed steps of ECMDF algorithm are given below:

The Training Phase:

Step 0. Read datasets of all sensors.

Step 1.1. Split datasets of sensors as training and test with user specified test size.

Step 1.2. Train separately both ANN and SVM for each sensor i.

Step 1.3. Probabilistically classify sensors' training datasets with trained and tuned ANN and SVM.

Step 1.4. For each sensor i, if ANN gives higher classification accuracy on training dataset of sensor i, then assign its probabilistic classification results for each proposition to respective BPA of sensor i for target k for the same proposition p, m_{kip}. Assign its probabilistic classification result to sensor i's piece of evidence for target k, m_{ki}. Assign its predicted classes of each target to sensor i's predicted classes for target k, mC_{ki}. Assign its classification accuracy ratio to sensor i's accuracy ratio, acc_i. Otherwise, assign SVM's results to them.

Step 1.5. To penalize the sensors that have lower sensor accuracies and to reward the ones that have higher accuracies, take the second power of each of them and normalize. Do it for each acc_i for each sensor i and update acc_i accordingly.

Step 2.1. By using sensor i's and sensor j's predicted classes for each target in training datasets, mC_{ki} and mC_{kj}, calculate the pairwise Pearson correlation coefficients, $corr_{ij}$. Built the correlation matrix, $CORR$.

$$corr_{ij} = \frac{\sum_{k=1}^{nbTrainingTargets}(mC_{ki} - \overline{mC}_{ki})(mC_{kj} - \overline{mC}_{kj})}{\sqrt{\sum_{k=1}^{nbTrainingTargets}(mC_{ki} - \overline{mC}_{ki})^2}\sqrt{\sum_{k=1}^{nbTrainingTargets}(mC_{kj} - \overline{mC}_{kj})^2}} \quad \forall i,j \tag{3.3}$$

$$CORR = \begin{bmatrix} corr_{11} & \cdots & corr_{1j} \\ \vdots & \ddots & \vdots \\ corr_{i1} & \cdots & corr_{ij} \end{bmatrix} \tag{3.4}$$

Where $\overline{mC}_{ki} = \frac{\sum_{l=1}^{nbTrainingTargets} mC_{li}}{nbTrainingTargets}$ and $\overline{mC}_{kj} = \frac{\sum_{l=1}^{nbTrainingTargets} mC_{lj}}{nbTrainingTargets}$.

Step 2.2. Scale each $corr_{ij}$ in correlation matrix according to the first interval in $sInterval$ and assign its result to each $sCorr_{ij}$ for sensor i and sensor j. The details of the scaling operation are given in Appendix A.

Step 2.3. For each target k, sensor i and sensor j; by using the scaled correlation coefficients in Step 2.2 as p parameter of L_p metric, calculate the distances between sensors i and j's evidences, m_{ki} and m_{kj}, for each target k by using sensors accuracies, acc_i and acc_j, as weight.

$$d_{kij} = \tag{3.5}$$

$$\left(\sum_{p=1}^{nbPropositions} (acc_i * acc_j)^{sCorr_{ij}} * |m_{kip} - m_{kjp}|^{sCorr_{ij}} \right)^{\frac{1}{sCorr_{ij}}} \quad \forall k,i,j$$

Step 2.4. Built distance matrix, DIS_k for each target k.

$$DIS_k = \begin{bmatrix} 1 & \cdots & d_{k1j} \\ \vdots & \ddots & \vdots \\ d_{ki1} & \cdots & 1 \end{bmatrix} \qquad \forall k \tag{3.6}$$

Step 2.5. Normalize each DIS_k for each target k.

Step 2.6. Calculate the similarity degrees between evidences of sensor i and j, s_{kij}, for each sensor and target k and built the similarity matrix for each target k.

$$s_{kij} = 1 - d_{kij} \qquad \forall k, i, j \tag{3.7}$$

$$SIM_k = \begin{bmatrix} 1 & \cdots & s_{k1j} \\ \vdots & \ddots & \vdots \\ s_{ki1} & \cdots & 1 \end{bmatrix} \qquad \forall k \tag{3.8}$$

Step 2.7. Calculate the credibility degrees, crd_{ki} of each sensor i's evidence for each target k. Then, take the averages of credibility degrees to find average credibility degree of sensor i for all targets.

$$sup_{ki} = \sum_{j=1}^{nbSensors} s_{kij} \qquad \forall k, i \tag{3.9}$$

$$crd_{ki} = \frac{sup_{ki}}{\sum_{j=1}^{nbSensors} sup_{kj}} \qquad \forall k, i \tag{3.10}$$

$$crd_i = \frac{\sum_{k=1}^{nbTrainingTargets} crd_{ki}}{nbTrainingTargets} \qquad \forall i \tag{3.11}$$

Step 2.8. To penalize the sensors that have lower credibility degrees and to reward the ones that have higher credibility degrees, take the second power of the calculated

credibility degrees in Step 2.7 in equation (3.11) and normalize them. Do it for each crd_i for each sensor i and update crd_i accordingly.

Step 2.9. Discount each BPA of sensor i for target k and proposition p, m_{kip}, and sum over all the sensors' to get one unified BPA , m'_{kp}, for each target k and proposition p.

$$m'_{kp} = \sum_{i=1}^{nbSensors} crd_i * m_{kip} \qquad \forall k, p \tag{3.12}$$

Step 2.10. Calculate the combined conflicting degree, k_{kt}, for each target k for combination step $t = 1$.

$$k_{kt} = \sum_{H_p \cap H_r = \emptyset} m'_{kp} * m'_{kr} \qquad \forall k, t = 1,2, \dots, nbSensors - 1 \tag{3.13}$$

Step 2.11. Using equation (3.14), combine the discounted BPA's once by using conflicting degree in (3.13). Then, go to Step 2.10, increase t by 1 and calculate the conflicting degree between the same discounted evidence by combining it t many times with itself. Through the conflicting degree, combine the discounted BPAs t many times. Repeat this steps $nbSensors - 1$ times for each target k and proposition p.

$$m''_{kpt} = \begin{cases} \dfrac{1}{1 - k_{kt}} \displaystyle\sum_{H_p \cap H_r = H_p} m'_{kp} * m'_{kr} \\ 0 \end{cases} \tag{3.14}$$

$$if\ p \neq \emptyset\ , \forall k, p, t = 1, \dots, nbSensors - 1$$
$$if\ p = \emptyset\ , \forall k, p, t = 1, \dots, nbSensors - 1$$

When $t = nbSensors - 1$, assign the value of m''_{kpt} to m''_{kp}.

Step 2.12. Calculate belief degree for each target k, and proposition p.

$$Bel(H_p)_k = \sum_{H_l \subset H_p} m''_{kl} \qquad \forall k, p \tag{3.15}$$

Step 2.13. Assign class of proposition p according to the highest belief degree, $Bel(H_p)_k$, to target k as the result of the fusion process.

Step 2.14. Turn back to Step 2.2. Scale correlation coefficient matrix, $CORR$ with the next interval in $sInterval$. Repeat the steps 2.2 to 2.13 until there is no new interval in $sInterval$.

Step 2.15. Choose the best scaling interval, u, that gives the highest number of correct predictions for test phase of ECMDF. Assign the credibility degrees calculated during best scaling interval to $fCrd_i$ to be used in test phase later.

The Test Phase:

Step 3.1. For each sensor, with the selected classifier in Step 1.4, probabilistically classify sensor i's test dataset. Assign its probabilistic classification results for each proposition to respective BPA of sensor i for target m for the same proposition p, mt_{mip}.

Step 3.2. With normalized credibility degrees determined in training phase in Step 2.15, $fCrd_i$, discount each BPA, mt_{mip}, for each sensor i, proposition p and target m and sum over all the sensors to get one unified BPA , mt'_{mp}, for each proposition p and target m.

$$mt'_{mp} = \sum_{i=1}^{nbSensors} fCrd_i * mt_{mip} \qquad \forall m, p \tag{3.16}$$

Step 3.3. Calculate combined conflicting degree, k_{mt}, for each target m for combination step $t = 1$.

$$k_{mt} = \sum_{H_p \cap H_r = \emptyset} mt'_{mp} * mt'_{mr} \qquad \forall m, t = 1,2, \dots, nbSensors - 1 \tag{3.17}$$

Step 3.4. Using equation (3.18), combine the unified BPA's once by using conflicting degree calculated in (3.17). Then, go to Step 3.3, increase t by 1 and calculate the conflicting degree between the same discounted evidence by combining it t many times with itself. Through the conflicting degree, combine the discounted BPAs t many times. Repeat this steps $nbSensors - 1$ times for each target m and proposition p.

$$mt''_{mpt} = \begin{cases} \dfrac{1}{1 - k_{mt}} \displaystyle\sum_{H_p \cap H_r = H_p} mt'_{mp} * mt'_{mr} \\ 0 \end{cases} \tag{3.18}$$

$$if\ p \neq \emptyset\ , \forall m, p, t = 1, \dots, nbSensors - 1$$
$$if\ p = \emptyset\ , \forall m, p, t = 1, \dots, nbSensors - 1$$

When $t = nbSensors - 1$, assign the value of mt''_{mpt} to mt''_{mp}.

Step 3.5. Calculate belief degree for each target m, and proposition p.

$$Bel(H_p)_m = \sum_{H_l \subset H_p} mt''_{ml} \qquad \forall m, p \tag{3.19}$$

Step 3.6. Assign the class of proposition p according to the highest belief degree, $Bel(H_p)_m$, to target m as the result of the fusion process.

3.7. Ensemble of Classifiers with Modified Distance Function and Sensor Accuracy (ECMDFS): Overview

In ECMDF, we discount BPAs by only credibility degrees of the sensors, which represents similarity of one sensor's evidences to other sensors'. This measure helps us to modify the evidences in such a way that if one or more of the sensors are poorly functioning, then its/their effect in the final combination of the evidences are decreased due to low correlation coefficients of sensors. However, if majority of sensors poorly function, then correlation between all of them may be high, which may lead to misleading results. To overcome this situation, we need another discounting factor. In the second approach, which is Ensemble of Classifiers with Modified Distance Function and Sensor Accuracy (ECMDFS), we discount the evidences in both training and test phase by sensor accuracy ratios and credibility degrees.

The flowchart in Figure 3.1 and in Figure 3.2 of ECMDF are valid for ECMDFS for training and test phases, respectively. Two approaches differ from each other in modifying the evidences. They also have the same objective function as equation (3.2). Detailed steps of ECMDFS algorithm is given in Appendix B.

CHAPTER 4

BENCHMARK MODELS

Our aim is to develop a new way of calculating evidence distances by L_p metric that allows us to overcome the conflicting and paradoxical situations of classical DST proposed by Dempster (1967) and Shafer (1976). The proposed algorithms aim to elastically calculate the evidence distances and assign credibility degrees of the sensors through them. Proposed algorithms, ECMDF and ECMDFS, are inspired from Yong et al.'s (2004) work in discounting evidences by credibility degrees. There are recent studies that generally utilize the idea in the seminal work of Yong et al. (2004). Therefore, in order to assess the performance of the proposed algorithms, we use the classical DST and Yong et al.'s (2004) works as our benchmark models. Since these benchmark models are used for fusion of information coming from multiple sensors, we use our framework intact except the fusion part. Thus, they only differ from ECMDF and ECMDFS in the fusion part of the evidences coming from multiple sensors. In this chapter, we give the details of them. In Section 4.1, Ensemble of Classifiers with Classical Dempster-Shafer Theory (ECDST), and in Section 4.2, Ensemble of Classifiers with Yong et al. (ECY) are given.

4.1. Ensemble of Classifiers with Classical Dempster-Shafer Theory (ECDST): Overview

The first benchmark model is the model that fuses probabilistic classification results with the classical DST. The flowchart of Ensemble of Classifiers with Classical Dempster-Shafer Theory (ECDST) is depicted in Figure 4.1.

ECDST starts with reading multi-sensor dataset in Step 0. In Step 1, each sensor's dataset is split as training and test. Then, with training dataset of all sensors, ANN and SVM is trained and tuned separately for each sensor. Like ECMDF and ECMDFS, based on the accuracy ratios, the one with the higher accuracy is chosen for each sensor. Then, test datasets of sensors are probabilistically classified with the chosen classifier. In Step 2, the model fuses the sensor classification results by the classical DST. The difference between ECDST and proposed algorithms is the fusion of the results in Step 2. ECMDF and ECMDFS combine evidences with the calculated credibility degrees in training phase whereas ECDST combines them according to the classical DST. The objective function of ECDST is the same as ECDMF and ECMDFS in equation (3.2), which is maximizing the number of correct predictions .

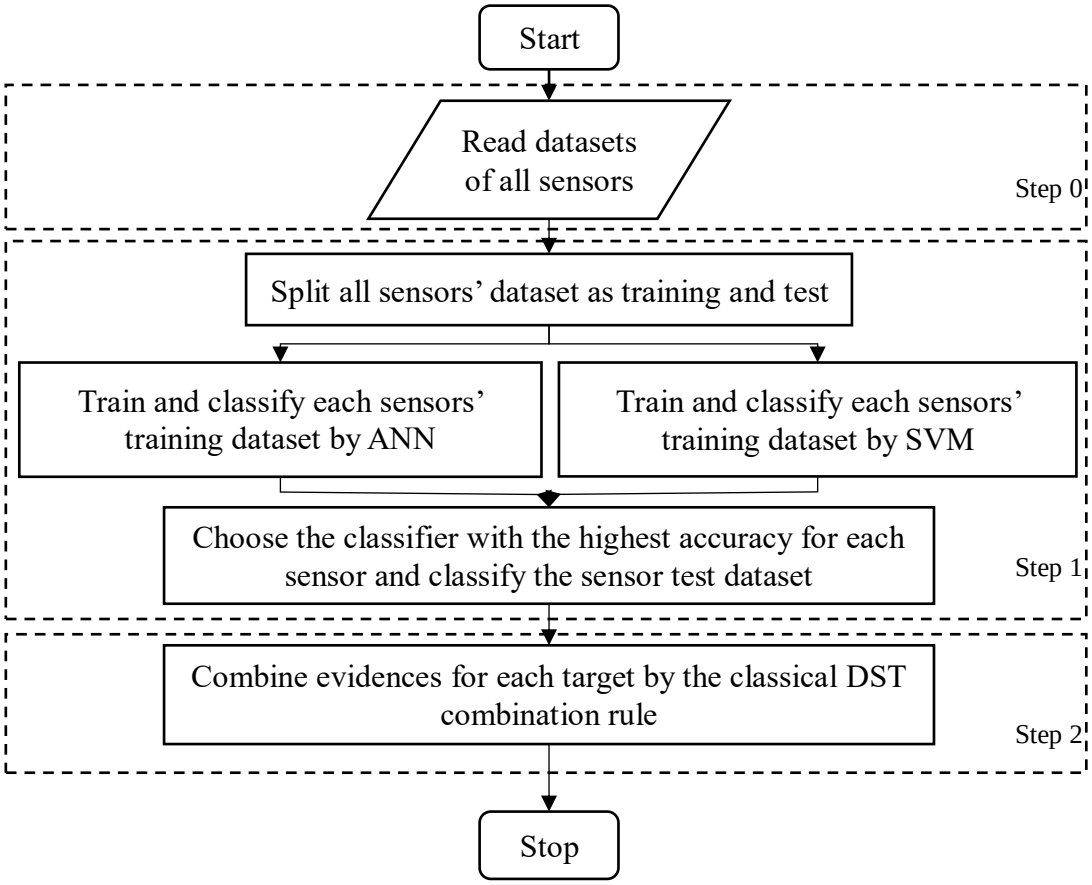

Figure 4.1. Flowchart of ECMDF

Details of ECDST is given below. We introduce the notations below in addition to the notations in Section 3.4.

k_{kij}	Combined conflicting degree of sensors between $1, 2, \dots, i-1, i$ and j for target k
m''_{mp}	Combined BPA for proposition p and target m

Steps of ECDST Algorithm

Step 0. Read datasets for all sensors.

Step 1.1. Split datasets of sensors as training and test with user specified test size.

Step 1.2. Train separately both ANN and SVM for each sensor i.

Step 1.3. Probabilistically classify sensors' training datasets with trained and tuned ANN and SVM.

Step 1.4. For each sensor i, if ANN gives higher classification accuracy on training dataset of sensor i, then, classify sensor i's test dataset with SVM. Assign its probabilistic classification results for each proposition to respective BPA of sensor i for target m for the same proposition p, m_{mip}. Otherwise, assign SVM's results to them.

Step 2.1. Calculate combined conflicting degree between sensors, k_{mij}, for each target m.

$$k_{mij} = \sum_{H_p \cap H_r = \emptyset} m_{mip} * m_{mir} \qquad \forall m, i, j \tag{4.1}$$

Step 2.2. Using equation (4.2), combine the evidences coming from the first and second sensor for proposition p and target m by using conflicting degree in (4.1). Then, go to Step 2.1, and calculate the conflicting degree between the first, second and third sensors, iteratively. Through the conflicting degree, combine the first

calculated combined evidence with the next sensor's evidence, third sensor's and update the combined evidence. Repeat this steps $nbSensors - 1$ time until there is no new evidence for the same target m for proposition p.

$$m''_{mp} = \begin{cases} \frac{1}{1 - k_{mij}} \sum_{H_p \cap H_r = H_p} m_{mip} * m_{mir} & if\ H_p \cap H_r \neq \emptyset \quad \forall m, p \\ 0 & if\ H_p \cap H_r = \emptyset \quad \forall m, p \end{cases} \tag{4.2}$$

Step 2.3. Calculate belief degree for each target m and proposition p.

$$Bel(H_p)_m = \sum_{H_l \subset H_p} m''_{ml} \qquad \forall m, p \tag{4.3}$$

Step 2.4. Assign the class of proposition p that has the highest belief degree, $Bel(H_p)_m$, to target m as the result of the fusion process.

Below, we provide an example that shows the calculations of Step 2.1 to Step 2.4.

Example: Suppose that there are three different evidences coming from three different sensors for the same target, which can be seen from Table 4.1.

Table 4.1. *Example evidences for ECDST for the same target*

Sensors	**Propositions**			
	A	**B**	**C**	**D**
1	0.5	0.2	0.3	0
2	0	0.9	0.1	0
3	0.55	0.1	0	0.35

Step 2.1. Calculate the conflicting degree between sensor 1 and 2.

Table 4.2. *Calculations for conflicting degree between first and second sensors*

Sensor 1-2	$m_{12A} = 0.00$	$m_{12B} = 0.90$	$m_{12C} = 0.10$	$m_{12D} = 0.00$
$m_{11A} = 0.50$	$\{A\}, 0.00$	$\{\emptyset\}, 0.45$	$\{\emptyset\}, 0.05$	$\{\emptyset\}, 0.00$
$m_{11B} = 0.20$	$\{\emptyset\}, 0.00$	$\{B\}, 0.18$	$\{\emptyset\}, 0.02$	$\{\emptyset\}, 0.00$
$m_{11C} = 0.30$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.27$	$\{C\}, 0.03$	$\{\emptyset\}, 0.00$
$m_{11D} = 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{D\}, 0.00$

$$k_{112} = 0.45 + 0.05 + 0.02 + 0.27 = 0.79 \tag{4.4}$$

Step 2.2. Combine the evidences coming from first and second sensor.

$$m''_{1A} = \frac{1}{(1 - 0.79)} * 0.00 = 0.00 \tag{4.5}$$

$$m''_{1B} = \frac{1}{(1 - 0.79)} * 0.18 = 0.86 \tag{4.6}$$

$$m''_{1C} = \frac{1}{(1 - 0.79)} * 0.03 = 0.14 \tag{4.7}$$

$$m''_{1D} = \frac{1}{(1 - 0.79)} * 0.00 = 0.00 \tag{4.8}$$

Go to Step 2.1, calculate the conflicting degree between first, second and third sensors.

Step 2.1. Calculate the conflicting degree between first, second and third sensor.

Table 4.3. *Calculations for conflicting degree between first and second sensors*

Sensor 1-2	$m_{12A} = 0.00$	$m_{12B} = 0.90$	$m_{12C} = 0.10$	$m_{12D} = 0.00$
$m_{11A} = 0.50$	$\{A\}, 0.00$	$\{\emptyset\}, 0.45$	$\{\emptyset\}, 0.05$	$\{\emptyset\}, 0.00$
$m_{11B} = 0.20$	$\{\emptyset\}, 0.00$	$\{B\}, 0.18$	$\{\emptyset\}, 0.02$	$\{\emptyset\}, 0.00$
$m_{11C} = 0.30$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.27$	$\{C\}, 0.03$	$\{\emptyset\}, 0.00$
$m_{11D} = 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{D\}, 0.00$

Table 4.4. *Calculations for conflicting degree between first, second and third sensors*

Sensor 1-2-3	$m_{13A} = 0.55$	$m_{13B} = 0.10$	$m_{13C} = 0.00$	$m_{13D} = 0.35$
$m''_{1A} = 0.00$	$\{A\}, 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$
$m''_{1B} = 0.86$	$\{\emptyset\}, 0.47$	$\{B\}, 0.09$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.30$
$m''_{1C} = 0.14$	$\{\emptyset\}, 0.08$	$\{\emptyset\}, 0.01$	$\{C\}, 0.00$	$\{\emptyset\}, 0.05$
$m''_{1D} = 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{\emptyset\}, 0.00$	$\{D\}, 0.00$

$$k_{113} = 0.47 + 0.30 + 0.08 + 0.01 + 0.05 = 0.91 \tag{4.9}$$

Step 2.2. Update the final combined evidences by combining the evidences coming from first, second and third sensor.

$$m''_{1A} = \frac{1}{(1 - 0.91)} * 0.00 = 0.00 \tag{4.10}$$

$$m''_{1B} = \frac{1}{(1 - 0.91)} * 0.09 = 1.00 \tag{4.11}$$

$$m''_{1C} = \frac{1}{(1 - 0.91)} * 0.00 = 0.00 \tag{4.12}$$

$$m''_{1D} = \frac{1}{(1 - 0.91)} * 0.00 = 0.00 \tag{4.13}$$

Step 2.3. Calculate the belief degree of each proposition.

$$Bel(H_A)_1 = \sum_{H_l \subset H_A} m''_{1l} = m''_{1A} = 0 \tag{4.14}$$

$$Bel(H_B)_1 = \sum_{H_l \subset H_B} m''_{1l} = m''_{1B} = 1 \tag{4.15}$$

$$Bel(H_C)_1 = \sum_{H_l \subset H_C} m''_{1l} = m''_{1C} = 0 \tag{4.16}$$

$$Bel(H_D)_1 = \sum_{H_l \subset H_D} m''_{1l} = m''_{1D} = 0 \tag{4.17}$$

Step 2.4. Proposition B has highest belief degree, $Bel(H_B)_1$, among the other ones. Thus, target is classified as type B.

4.2. Ensemble of Classifiers with Yong et al. (ECY): Overview

The second benchmark model fuses sensor probabilistic classification results with the modified DST proposed by Yong et al. (2004). The flowchart of the Ensemble of Classifiers with Yong et al. (ECY) is given in Figure 4.2.

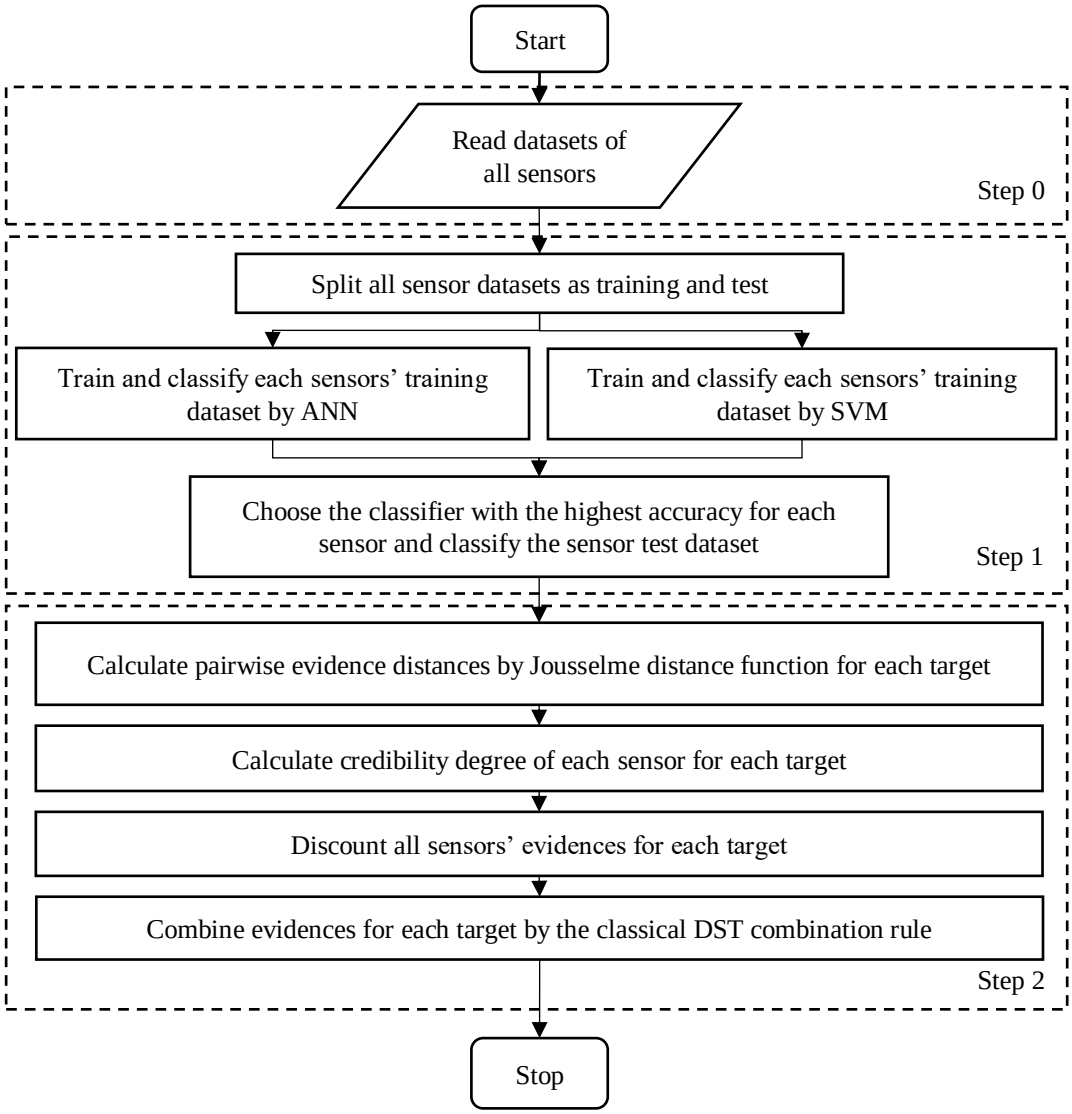

Figure 4.2. Flowchart of ECY

Step 0 and Step 1 of the ECY are exactly the same with the Step 0 and Step 1 of ECDST. After each sensor's test data is probabilistically classified in Step 1, evidence distances are calculated according to Jousselme distance for each target. Based on the distances, credibility degrees are calculated for each target. Then, probabilistic classification results are discounted with credibility degrees and combined with classical DST. Finally, belief degree of each proposition for each target is calculated and the final class of the targets is decided. ECY has the same objective function same as ECDMF, ECMDFS and ECDST in equation (3.2),

Below, details of ECY is given. The same notations in Section 3.4 are used.

Steps of ECY Algorithm

Step 0 and Step 1. Same with ECDST.

Step 2.1. Calculate pairwise distances between evidences, d_{mij}, according to the Jousselme distance function for each target m.

$$d_{mij} = \sqrt{\frac{1}{2}(\overrightarrow{mt_{mi}} - \overrightarrow{mt_{mj}})^T * D * (\overrightarrow{mt_{mi}} - \overrightarrow{mt_{mj}})} \qquad \forall m, i, j \tag{4.18}$$

where D is a matrix whose elements are $|H_p \cap H_r|/|H_p \cup H_r|$ and $|.|$ is the cardinality.

Step 2.2. Calculate distance matrix, DIS_m for each target m.

$$DIS_m = \begin{bmatrix} 1 & \cdots & d_{m1j} \\ \vdots & \ddots & \vdots \\ d_{mi1} & \cdots & 1 \end{bmatrix} \qquad \forall m \tag{4.19}$$

Step 2.3. Normalize each DIS_m for each target m.

Step 2.4. Calculate the similarity degrees, s_{mij}, between each sensor for each target m.

$$s_{mij} = 1 - d_{mij} \qquad \forall m, i, j \tag{4.20}$$

Step 2.5. Calculate the credibility degrees, crd_{mi}, for each sensor i and target m.

$$sup_{mi} = \sum_{j=1}^{nbSensors} s_{mij} \qquad \forall m, i \tag{4.21}$$

$$crd_{mi} = \frac{sup_{mi}}{\sum_{j=1}^{nbSensors} sup_{mj}} \qquad \forall m, i \tag{4.22}$$

Step 2.6. Discount each BPA for each sensor i, proposition p and target m, mt_{mip}, and sum over all the sensors to get one unified BPA , mt'_{mp}, for each proposition p and target m.

$$mt'_{mp} = \sum_{i=1}^{nbSensors} crd_{mi} * mt_{mip} \qquad \forall m, p \tag{4.23}$$

Step 2.7. Calculate combined conflicting degree, k_{mt}, for each target m for combination step $t = 1$.

$$k_{mt} = \sum_{H_p \cap H_r = \emptyset} mt'_{mp} * mt'_{mr} \qquad \forall m, t = 1,2, \dots, nbSensors - 1 \tag{4.24}$$

Step 2.8. Using equation (4.25), combine the unified BPA's once by using conflicting degree in (4.24). Then, go to Step 2.7, increase t by 1 and calculate the conflicting degree between the same discounted evidence by combining it t many times with itself. Through the conflicting degree, combine the discounted BPAs t many times. Repeat this steps $nbSensors - 1$ times for each target m and proposition p.

$$mt''_{mpt} = \begin{cases} \dfrac{1}{1-k_{mt}} \displaystyle\sum_{H_p \cap H_r = H_p} mt'_{mp} * mt'_{mr} & if\ p \neq \emptyset\ , \forall m, p, t = 1, \dots, nbSensors - 1 \\ 0 & if\ p = \emptyset\ , \forall m, p, t = 1, \dots, nbSensors - 1 \end{cases} \quad (4.25)$$

When $t = nbSensors - 1$, assign the value of mt''_{mpt} to mt''_{mp}.

Step 2.9. Calculate belief degree for each target m and proposition p.

$$Bel(H_p)_m = \sum_{H_l \subset H_p} mt''_{ml} \qquad \forall m, p \quad (4.26)$$

Step 2.10. Assign the class of proposition p that has the highest belief degree, $Bel(H_p)_m$, to target m as the result of the fusion process.

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

In this chapter, we present the computational results of the proposed algorithms, and compare them with benchmark models. In Section 5.1, we give the details of the datasets used. In Section 5.2, we present parameter settings. In Section 5.3, we introduce the performance measures. In Section 5.4, we give the computational results. Dataset generation algorithms for paradoxical situations are given in Section 5.5. Finally, computational results for paradoxical situations are given in Section 5.6.

5.1. Datasets

To assess the performance of the proposed algorithms, numerical multi-sensor dataset is needed. However, finding unclassified real data in sensor data fusion is problematic. Thus, we generated numerical multi-sensor artificial datasets by using a method adopted from Guyon (2003). This method allows us to generate classification dataset together with corresponding class labels with user specified parameters.

While generating the datasets, we used the algorithm depicted in Figure 5.1. According to the algorithm, firstly, the number of sensors is determined. Then, classification dataset by adopting Guyon's (2003) algorithm is generated. Next, this dataset is replicated the number of sensor times. To make the sensors dataset different from each other, noise generated according to normal distribution with mean 0 and variance 0.05 is added to each sensors' dataset. Here, variance in normal distribution is kept small due to avoid large deviations from the mean and make the datasets more consistent. While adding the noise, we used the algorithm proposed by Xu and Yu (2017), which is given in equation (5.1).

h: Index for features

$a_{0h}(x_k)$: Original h^{th} feature value for target k

$a_{ih}(x_k)$: Sensor i's h^{th} feature value for target k

w_i: Noise value generated according to $N(0, 0.05)$ for sensor i

$$a_{ih}(x_k) = a_{0h}(x_k) * (1 + w_i) \tag{5.1}$$

In real ATR systems, each radar may provide different feature readings about the same potential target. Then, that information from each radar are combined to predict the potential target's class. To apply the same concept to all of the models in our study, some of the features determined randomly are dropped from each sensors' dataset to represent the heterogenous sensors.

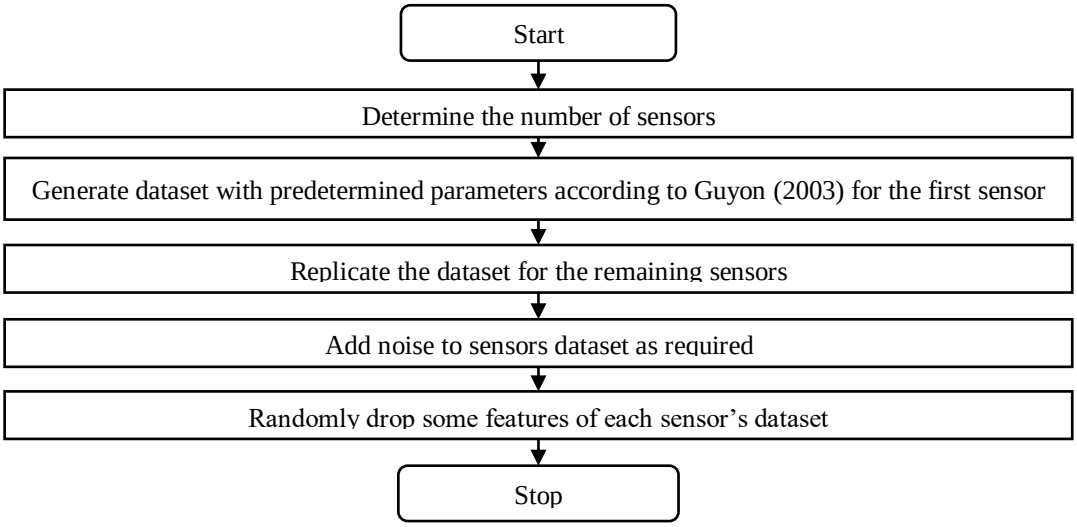

Figure 5.1. Dataset generation algorithm

We test the algorithms with small, medium and large datasets. Thus, we used three different levels for the number of samples, which are 100, 500 and 1000. Two different levels for the number of features are used, i.e. 5 and 10. Three different settings for

the number of noisy sensors are used, i.e. 0, 1 and 2. When the number of noisy sensors is 0, all sensors are consistent. When the number of noisy sensors is greater than 0, given number of sensors are not consistent with the rest of the sensors. There is a total of 18 different problem settings as shown in Table 5.1. For each problem setting, 5 different datasets are generated with different seed values.

Table 5.1. *Summary of the problem settings*

Problem Setting[1]	**Number of Features**	**Number of Samples**	**Number of Noisy Sensors**
5/100/0	5	100	0
5/500/0	5	500	0
5/1000/0	5	1000	0
5/100/1	5	100	1
5/500/1	5	500	1
5/1000/1	5	1000	1
5/100/2	5	100	2
5/500/2	5	500	2
5/1000/2	5	1000	2
10/100/0	10	100	0
10/500/0	10	500	0
10/1000/0	10	1000	0
10/100/1	10	100	1
10/500/1	10	500	1
10/1000/1	10	1000	1
10/100/2	10	100	2
10/500/2	10	500	2
10/1000/2	10	1000	2

[1] Number of features, number of samples and number of noisy sensors are represented by x, y and z in *Problem Setting* column by x/y/z, respectively.

If the number of noisy sensors is greater than 0, another dataset with additional parameters is generated with Guyon's (2003) algorithm for noisy sensors. In this

dataset, all class labels of noisy sensors are randomly changed. In addition, feature values of each noisy sensor are shifted by a given number to make the classification task even harder. For the experimental purposes, feature values of noisy sensors' are shifted by 3.

Figures 5.2 and 5.3 show the samples of generated datasets for the same problem setting. They are generated with 2 feature values, 1000 samples and 3 classes. In both of the figures, each color represents one of the three classes. Horizontal and vertical axes show the feature values. From Figure 5.2, it can be seen that classes are separated clearly. On the other hand, in Figure 5.3, each observations' feature values are shifted around 3, and their classes, i.e. colors, are randomly exchanged. By doing so, we make the noisy sensors inconsistent with the consistent sensors in the ensemble.

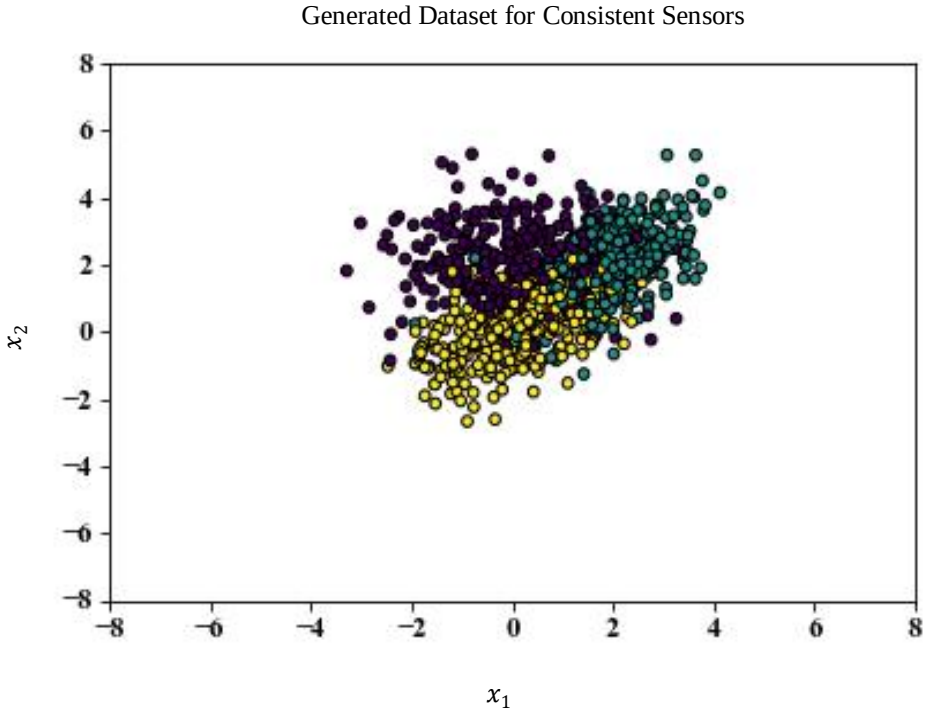

Figure 5.2. Example of generated dataset for consistent sensors

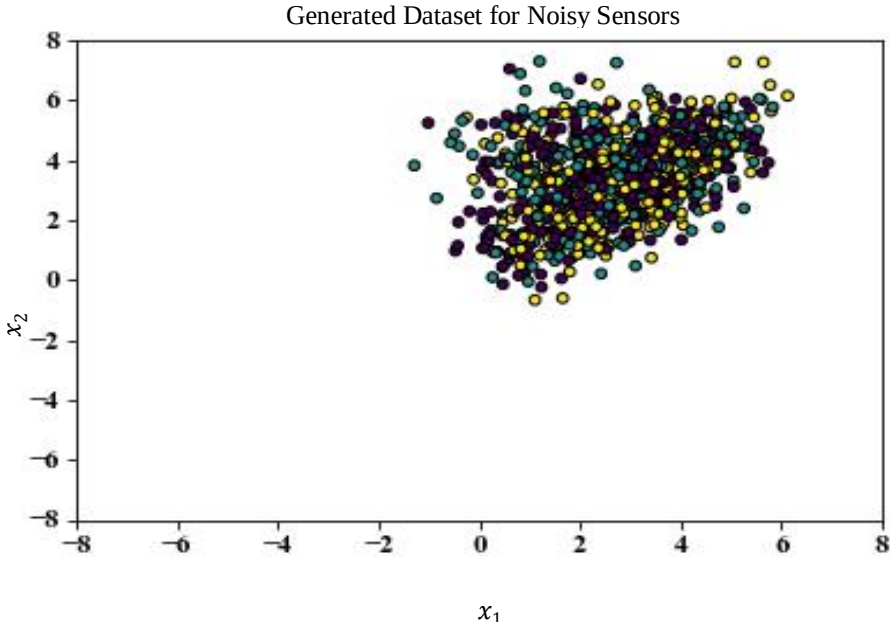

Figure 5.3. Example of generated dataset for noisy sensors

5.2. Parameter Settings

In Table 5.2, parameter settings are given. We choose to use 3 classes in our experiments. In ECMDF, ECMDFS and the benchmark models, we train both ANN and SVM for each sensor. Thus, as the number of sensors increase, the training time of each algorithm increases substantially. Thus, in order to keep the training time reasonable, we preferred to use 3 sensors in each problem settings. We test our algorithms for a range of values representing the full spectrum of the L_p metric. [1,2], [1,5], [1,10], [1,20], [1,40] are used as scaling intervals. While training both ANN and SVM, 2/3 of the generated dataset is used as the training data in each problem settings.

Table 5.2. *Parameter settings for all ensembles*

Parameters	Value
Number of classes	3
Number of sensors	3
Scaling interval	[1,2], [1,5], [1,10], [1,20], [1,40]
Training size	2/3

For setting the hyper parameters of ANN and SVM, random search and grid search are used, respectively. Tuning operations of ANN and SVM are performed after training. Both are trained with the default parameters given in Appendix C. To tune both ANN and SVM, different combinations of hyper parameters are passed to the models, which are given in Table 5.3 for ANN and in Table 5.4 for SVM. The final values of activation function, optimizer, batch size and number of epochs for ANN, and kernel function, C and degree, if polynomial kernel function is used, for SVM are decided based on the generated dataset dynamically. Tuning operation is performed with training data.

Table 5.3. *Hyper parameters tuned with random search and their passed combinations for ANN*

Hyper Parameters	Values/Types
Activation Function	Softmax activation function
	Rectified linear unit
	Hyperbolic tangent activation function
	Sigmoid activation function
Optimizer	Stochastic gradient descent
	Root mean square propagation
	Adaptive gradient
	Adaptive moment estimation
Batch Size	32 or number of samples
Epoch	[10,20,30,40,50]

Table 5.4. *Hyper parameters tuned with grid search and their passed combinations for SVM*

Hyper Parameters	Values/Types
Kernel Function	Radial basis function
	Polynomial kernel function
C	[1, 5, 10]
Degree for Polynomial Kernel Function	[1, 2, 3]

Details of the hyper parameters of ANN and SVM are given in Appendix D and E, respectively.

If the number of hidden layers increases, we may end up with the overfitting problem. In this situation, ANN may lose its generalization capability. Thus, for each sensors' ANN, one hidden layer is used.

The number of neurons in each hidden layer is an important decision for performance of ANN. The method proposed by Ke and Liu (2008) is recommended for minimizing mean square error (MSE) (Sheela and Deepa 2013). Thus, we adopted Ke and Lie (2008) method that is given below.

n: Number of neurons in hidden layer

L: Number of hidden layer

f: Number of features

s: Number of samples

$$n = \frac{f + \sqrt{s}}{L} \tag{5.2}$$

In order to get probabilistic classification results, we use categorical cross entropy loss function and softmax activation function in the output layer of ANN. By random search, activation function for the input and hidden layer are chosen.

To determine the learning rate and number of iterations for random search, we performed design of experiments given in Table 5.5.

Table 5.5. *Design of experiments for learning rate and number of iterations for random search*

Problem Setting	**Number of Features**	**Number of Samples**	**Number of Iterations for Random Search**	**Learning Rate**
5/100/5/0.001	5	100	5	0,001
5/1000/5/0.001	5	1000	5	0,001
10/100/10/0.001	10	100	10	0,001
10/1000/10/0.001	10	1000	10	0,001
5/100/5/0.01	5	100	5	0,01
5/1000/5/0.01	5	1000	5	0,01
10/100/10/0.01	10	100	10	0,01
10/1000/10/0.01	10	1000	10	0,01
5/100/5/0.1	5	100	5	0,1
5/1000/5/0.1	5	1000	5	0,1
10/100/10/0.1	10	100	10	0,1
10/1000/10/0.1	10	1000	10	0,1

[1] Number of features, number of samples, number of iterations for random search and learning rate are represented by x, y, z, w in *Problem Setting* column by x/y/z/w, respectively.

Single problem is solved for each of the problem setting in Table 5.5 and results are shown in Figures 5.4, 5.5 and 5.6. In Figure 5.4, accuracy ratio is 0.001. In this situation, maximum accuracy ratio is approximately 0.8 in dataset 10/1000/10/0.001.

From Figure 5.5, we can see the accuracy ratios for learning rate 0.01 with different number of features, number of samples and number of iterations for random search. Problem setting 10/1000/10/0.01 gives the highest accuracy ratio as approximately 0.9 for learning rate being equal to 0.01. Finally, from Figure 5.6, we can see the accuracy ratios when the learning rate is equal to 0.1. It is clear that ANN gives the highest accuracy ratio among the other learning rates when the learning rate is 0.1. Thus, we set the learning rate as 0.1 for our problem setting in ensemble.

We may also realize that when number of epochs is greater than 50, accuracy ratio does not change too much. For this reason, number of epochs in random search is restricted to values between 10 and 50.

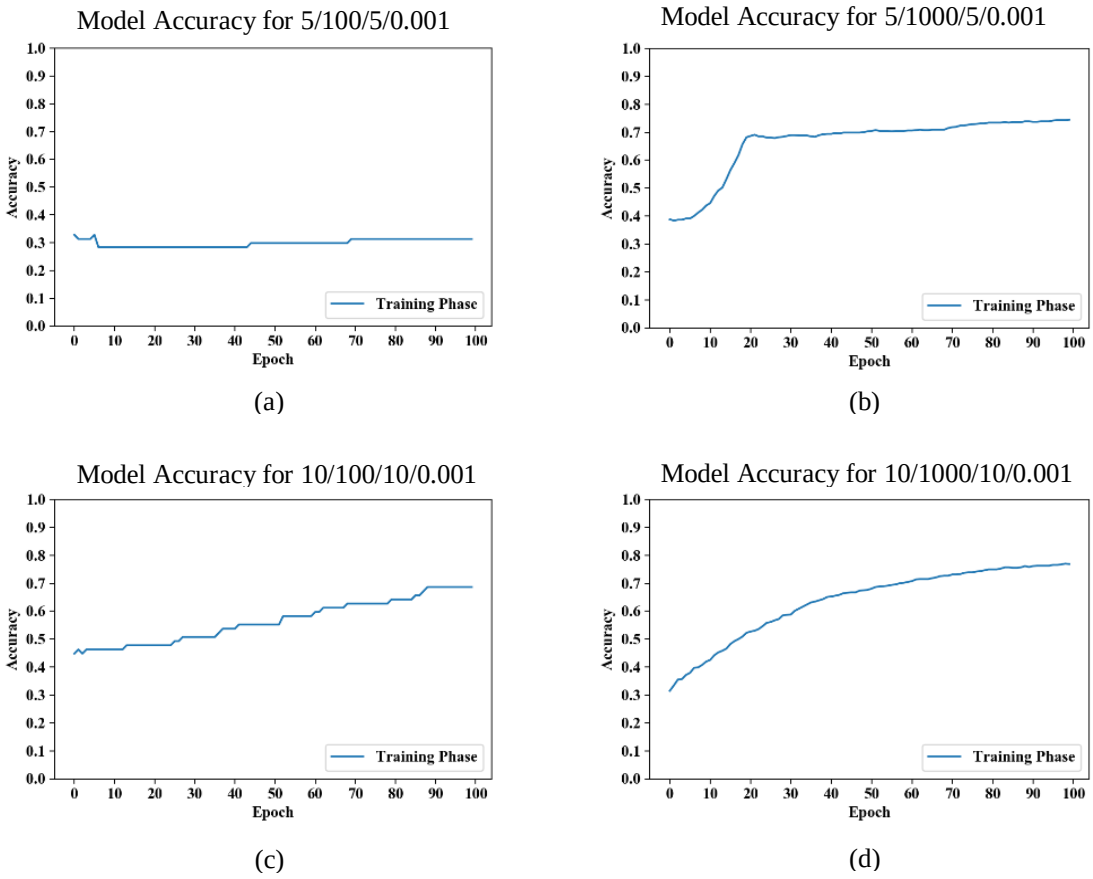

Figure 5.4. Model accuracy in each epoch when learning rate is 0.001

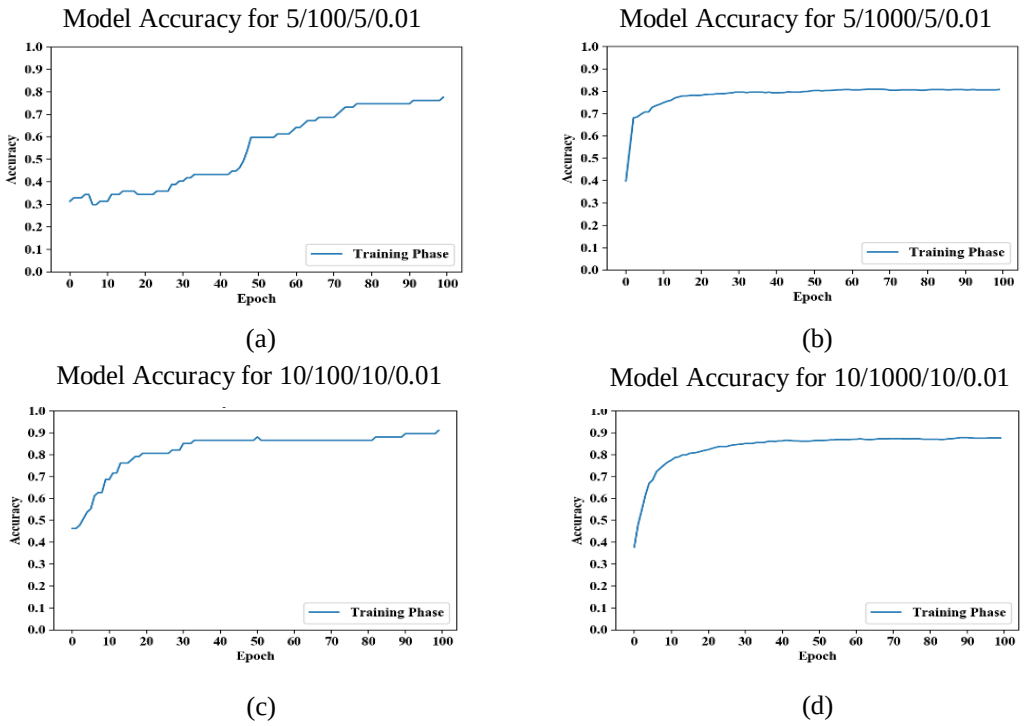

Figure 5.5. Model accuracy in each epoch when learning rate is 0.01

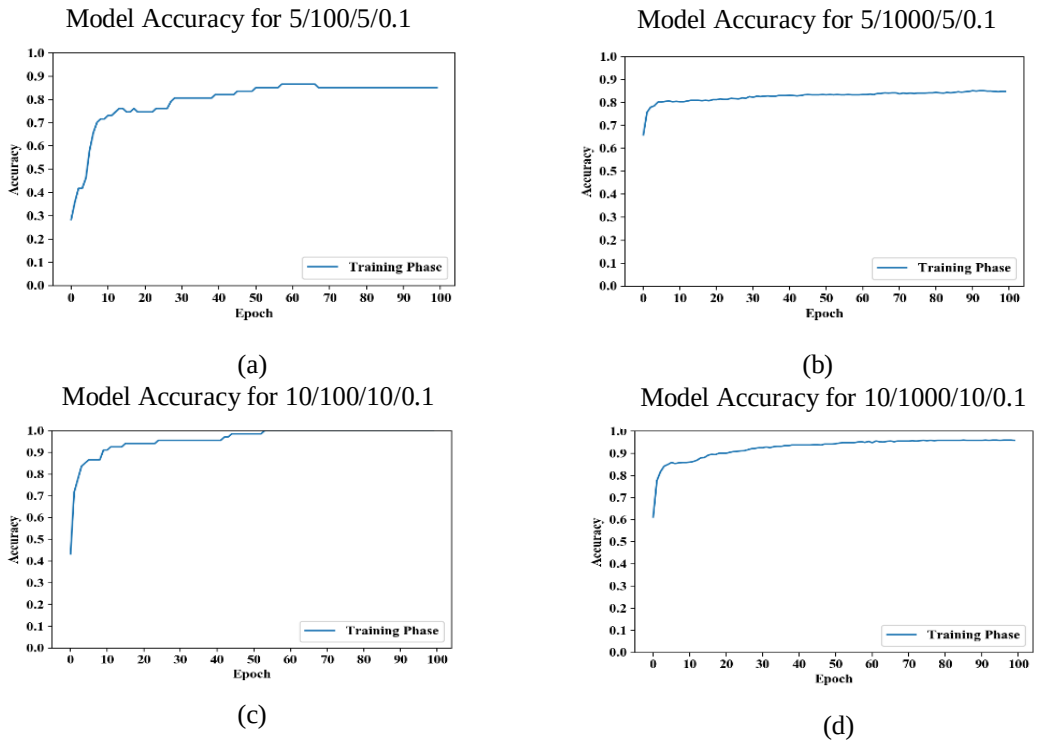

Figure 5.6. Model accuracy in each epoch when learning rate is 0.1

Table 5.6 shows the computational times of each problem settings in Table 5.5. When the number of iterations for random search is 10, the training times of the ANN doubles. Thus, we set the level of it as 5.

Table 5.6. *Computational times for design of experiment for learning rate and number of iterations in random search*

Problem Setting[1]	**Training Time (sec)**
5/100/5/0.001	18
5/1000/5/0.001	25
10/100/10/0.001	47
10/1000/10/0.001	54
5/100/5/0.01	18
5/1000/5/0.01	25
10/100/10/0.01	49
10/1000/10/0.01	56
5/100/5/0.1	20
5/1000/5/0.1	29
10/100/10/0.1	44
10/1000/10/0.1	54

[1] Number of features, number of samples, number of iterations for random search and learning rate are represented by x, y, z, w in *Problem Setting* column by x/y/z/w, respectively.

5.3. Performance Measures

To evaluate the classification quality of the proposed algorithms, we use four different performance measures, which are percent of correct predictions (PCP), average probability change in all targets (AC), average probability change in correctly predicted targets (CC) and average probability change in wrong predictions (WC).

Before introducing the performance measures in detail, we give some definitions to make things clearer. *True proposition* represents the real class of each target. Prediction of the ensemble is said to be c*orrect prediction* if it correctly identifies the real class of the target. On the other hand, prediction is said to be *wrong prediction* if ensemble does not correctly identifies the real class of the target.

For example; Suppose that there are three different propositions in FOD as $\{A, B, C\}$. Each target is either type A, B or C. Suppose also that class of the one of the target is A. That is the *true proposition* for this target is A. If the model predicts its class as A, then this prediction is said to be *correct prediction*. On the other hand, if the model predicts its class as B, then this prediction is said to be *wrong prediction*.

Percent of correct predictions (PCP): This performance metric measures the percent of correct predictions over the test dataset. As it gets higher, classification power of the model increases. It is calculated according to (5.4).

$$y_m = \begin{cases} 1 & \text{if class of target } m \text{ is correctly predicted} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$PCP = \frac{\sum_{m=1}^{n} y_m}{nbTestTargets} \quad \text{where } m = 1,2,\dots,nbTestTargets \tag{5.4}$$

Average probability change in all targets (AC): It measures average probability change in true propositions probability between average probabilities coming from ANN or SVM for each target and sensor, and final combined evidences for all targets. It shows how the model changes the probability of true propositions belief degree in

each target. Positive AC shows that the model is effective in raising the true proposition's belief degree. Negative AC shows that the model decreases the belief degrees of true propositions on the average.

$$h_{mp} = \begin{cases} 1 \text{ if } p \text{ is the true proposition for target } m \\ 0 \text{ otherwise} \end{cases} \quad (5.5)$$

$$aver_{mp} = \frac{\sum_{i=1}^{nbSensors} mt_{mip}}{nbSensors} \qquad \forall m, p \quad (5.6)$$

$$C = \sum_{m=1}^{nbTestTargets} \sum_{p=1}^{nbProposition} \left(\left(mt''_{mp} - aver_{mp} \right) * h_{mp} \right) \quad (5.7)$$

$$AC = \frac{C}{nbTestTargets} \quad (5.8)$$

Average probability change in correctly predicted targets (CC): It measures average probability change in correct predictions' belief degrees between average probabilities coming from ANN or SVM for each target and sensor, and final combined evidences for correct predictions. It shows how the model changes the probability of correct predictions' belief degrees on the average in each target.

$$aver_{mp} = \frac{\sum_{i=1}^{nbSensors} mt_{mip}}{nbSensors} \qquad \forall m, p \quad (5.9)$$

$$C = \sum_{m=1}^{nbTestTargets} \sum_{p=1}^{nbProposition} \left(\left(mt''_{mp} - aver_{mp} \right) * y_m \right) \quad (5.10)$$

$$CC = \frac{C}{nbTestTargets} \quad (5.11)$$

Average probability change in wrong predictions (WC): It measures average probability change in wrong predictions' belief degrees between average probabilities coming from ANN or SVM for each target and sensor, and final combined evidences for wrong predictions. It shows how the model changes the probability of wrong predictions' belief degrees on the average in each target.

$$aver_{mp} = \frac{\sum_{i=1}^{nbSensors} mt_{mip}}{nbSensors} \qquad \forall m, p \tag{5.12}$$

$$C = \sum_{m=1}^{nbTestTargets} \sum_{p=1}^{nbProposition} \left(\left(mt''_{mp} - aver_{mp} \right) * (1 - y_m) \right) \tag{5.13}$$

$$WC = \frac{C}{nbTestTargets} \tag{5.14}$$

5.4. Computational Results

All models are coded in Python and computational experiments are conducted in a PC with Intel ® Core ™ i5-5200U 2.20 GHz processor, 8 GR RAM, 64 Bit OS with Windows 10 operating system.

ECDST, ECY, ECMDF and ECMDFS are solved for 18 different problem settings in Table 5.1. In each problem setting, 5 different datasets are generated with different seed values and summary of the results are presented in Tables 5.7, 5.8, 5.9 and 5.10. The detailed experimental results are presented in Appendix F for each dataset for each problem settings.Table 5.7 shows the average results of five different datasets for each problem setting. In the last row of Table 5.7, we also give the overall averages for all of the 90 datasets for each ensemble.

Table 5.7. *Summary of PCP values for each problem setting*

Problem Setting	**PCP (%)**			
	ECDST	**ECY**	**ECMDF**	**ECMDFS**
5/100/0	81.82	81.82	81.82	81.82
5/500/0	91.15	90.30	90.30	90.42
5/1000/0	88.73	85.64	88.24	88.24
5/100/1	80.61	77.58	81.21	81.82
5/500/1	89.45	84.36	87.88	88.85
5/1000/1	86.91	79.09	85.03	85.76
5/100/2	72.73	39.39	43.64	73.33
5/500/2	85.21	33.45	64.97	85.82
5/1000/2	82.42	35.52	79.03	82.61
10/100/0	77.58	77.58	77.58	77.58
10/500/0	88.97	87.27	88.24	88.73
10/1000/0	89.09	86.97	88.42	88.30
10/100/1	72.12	64.24	72.12	73.94
10/500/1	87.39	80.12	82.42	86.42
10/1000/1	86.36	80.73	83.09	85.21
10/100/2	69.70	47.88	49.70	61.82
10/500/2	81.21	41.58	57.33	80.97
10/1000/2	83.82	37.52	63.58	84.30
Overall	83.07	67.28	75.81	82.55

*Values in the table are the averages of the results of five different datasets for each problem setting.

We firstly compare ECY, ECMDF and ECMDFS with ECDST. Out of 18 different problem settings, ECY and ECMDF give worse performance than ECDST in 16 and 14 problem settings, respectively. ECY gives the same results with ECDST for 2 of the problem settings, 5/100/0 and 10/100/0. On the other hand, ECMDF gives the same results with ECDST for 3 of the problem settings, and gives higher result than ECDST for only one of the problem settings, 5/100/1 with 0.61% difference. It can be seen from Table 5.7 that when the number of noisy sensors is greater than 0, ECY and ECMDF begin to perform poorly due to the nature of the calculation of credibility degrees. Credibility degrees show the similarity between each sensor and other sensors. When the number of noisy sensors in the ensemble is greater than the number of consistent sensors, credibility degrees of noisy sensors begin to increase. This situation reduces the consistent sensors' evidences and increases the effect of noisy

sensors' evidences in the evidence combination phase in each target. Thus, the difference between PCP values of ECY and ECMDF from ECDST is much higher in problem settings 5/100/2, 5/500/2, 5/1000/2, 10/100/2, 10/500/2 and 10/1000/2. The maximum difference between PCP values of ECY and ECDST occurs in 5/500/2 with -54.76% (33.45%-85.21%). For the same problem setting, the difference between PCP values of ECMDF and ECDST is -20.24% (64.97%-85.21%). Also, ECMDF's worst performance occurs in 5/100/2 with -29.09% (43.64%-72.73%).

Since ECMDFS discounts the evidences by both credibility degrees and sensor accuracies, it gives better PCP results than ECY and ECMDF when the number of noisy sensors is greater than 0. Out of 18 problem settings, ECMDFS gives better PCP results in 6 problem settings than ECDST, and gives the same results in 2 problem settings. We get the highest difference between PCP values in 5/100/1 and 10/100/1, which are 1.21% and 1.82%, respectively. Also, when the number of features is 5 and number of noisy sensors is 2, ECMDFS's PCP values are higher than ECDST for all sizes of datasets. The worst performance of ECMDFS occurs in problem setting 10/100/2. For this setting, the difference between PCP values of ECMDFS and ECDST is -7.88% (61.82%-69.70%).

In order to test the statistical significance, we performed paired t-test. Table 5.8 shows the related p values of hypothesis testing of PCP values of each ensemble and ECDST. At 5% significance level, ECDST shows better PCP values than ECY and ECMDF. There is no statistically significant difference between ECDST and ECMDFS.

Table 5.8. *p values of hypothesis testing for PCP values between each ensemble and ECDST*

Ensemble	**ECY**	**ECMDF**	**ECMDFS**
ECDST	0.0023	0.0074	0.2883

In Table 5.9, we also give the p values of paired t-tests between PCP values of ECY and other ensembles. At 5% significance level, all of the ensembles give better PCP values than ECY.

Table 5.9. *p values of hypothesis testing for PCP values between each ensemble and ECY*

Ensemble	ECDST	ECMDF	ECMDFS
ECY	0.0023	0.0102	0.0032

We report the summary of AC, CC and WC values for each problem setting in Tables 5.10, 5.11 and 5.12, respectively.

Table 5.10. *Summary of AC values for each problem setting*

Problem Setting	AC			
	ECDST	ECY	ECMDF	ECMDFS
5/100/0	0.1704	0.1531	0.1606	0.1638
5/500/0	0.1513	0.1379	0.1312	0.1371
5/1000/0	0.1506	0.1222	0.1322	0.1376
5/100/1	0.2226	0.2005	0.1759	0.2323
5/500/1	0.2551	0.1886	0.2027	0.2439
5/1000/1	0.2552	0.1795	0.1839	0.2399
5/100/2	0.1859	-0.0694	-0.0362	0.1707
5/500/2	0.2856	-0.1384	-0.0421	0.2462
5/1000/2	0.2729	-0.1246	-0.0257	0.2409
10/100/0	0.1461	0.1248	0.1336	0.1363
10/500/0	0.1698	0.1513	0.1576	0.1612
10/1000/0	0.1434	0.1312	0.1351	0.1357
10/100/1	0.1563	0.1143	0.1351	0.1520
10/500/1	0.2545	0.1617	0.1707	0.2331
10/1000/1	0.2352	0.1776	0.1876	0.2273
10/100/2	0.1863	-0.0024	-0.0012	0.1098
10/500/2	0.2507	-0.0837	-0.0301	0.2118
10/1000/2	0.2825	-0.1224	-0.0357	0.2413
Overall	0.2097	0.0723	0.0964	0.1900

*Values in the table are the averages of the results of five different datasets for each problem setting.

Table 5.11. *Summary of CC values for each problem setting*

Problem Setting	**CC**			
	ECDST	**ECY**	**ECMDF**	**ECMDFS**
5/100/0	0.2387	0.2196	0.2256	0.2297
5/500/0	0.1790	0.1765	0.1596	0.1655
5/1000/0	0.1890	0.1765	0.1677	0.1746
5/100/1	0.2976	0.2837	0.2356	0.3124
5/500/1	0.2987	0.2444	0.2439	0.2896
5/1000/1	0.3108	0.2573	0.2332	0.2986
5/100/2	0.2754	0.1025	0.1038	0.2503
5/500/2	0.3415	-0.0234	0.0013	0.2959
5/1000/2	0.3442	-0.0636	-0.0236	0.3096
10/100/0	0.2237	0.1944	0.2046	0.2076
10/500/0	0.2086	0.1979	0.1939	0.1981
10/1000/0	0.1783	0.1780	0.1672	0.1686
10/100/1	0.2509	0.1999	0.2122	0.2380
10/500/1	0.3085	0.2356	0.2233	0.2871
10/1000/1	0.2926	0.2508	0.2476	0.2870
10/100/2	0.2963	0.1392	0.1235	0.2236
10/500/2	0.3228	0.0469	0.0408	0.2795
10/1000/2	0.3549	-0.0224	0.0081	0.3059
Overall	0.2729	0.1552	0.1538	0.2512

*Values in the table are the averages of the results of five different datasets for each problem setting.

From Table 5.10, we can see that ECY and ECMDF lower the probability of true propositions belief degrees (AC) when the number of noisy sensors is equal to 2. In addition, Table 5.11 shows that ECY and ECMDF also lowers CC in problem settings 5/500/2, 5/1000/2, 10/1000/2, and 5/1000/2, respectively. On the other hand, ECDST and ECMDFS are promising in a sense that both of them are successful in increasing the probability in AC and CC. However, all of the ensembles decrease WC for all of the problem settings as can be seen from Table 5.12. That is they lower the true propositions belief degree in situations that they wrongly predicted the class of the targets.

Table 5.12. *Summary of WC values for each problem setting*

Problem Setting	**WC**			
	ECDST	**ECY**	**ECMDF**	**ECMDFS**
5/100/0	-0.1357	-0.1567	-0.1387	-0.1370
5/500/0	-0.1335	-0.2098	-0.1303	-0.1264
5/1000/0	-0.1419	-0.1856	-0.1218	-0.1281
5/100/1	-0.0782	-0.0932	-0.0763	-0.1193
5/500/1	-0.1064	-0.1256	-0.1079	-0.1124
5/1000/1	-0.1152	-0.1170	-0.0958	-0.1141
5/100/2	-0.0445	-0.1852	-0.1107	-0.0388
5/500/2	-0.0763	-0.1955	-0.1013	-0.0986
5/1000/2	-0.0657	-0.1578	-0.0320	-0.0924
10/100/0	-0.1217	-0.1264	-0.1068	-0.1067
10/500/0	-0.1462	-0.1671	-0.1102	-0.1248
10/1000/0	-0.1418	-0.1802	-0.1110	-0.1129
10/100/1	-0.0947	-0.1025	-0.1086	-0.1220
10/500/1	-0.1206	-0.1415	-0.1072	-0.1212
10/1000/1	-0.1277	-0.1251	-0.1158	-0.1174
10/100/2	-0.0970	-0.1294	-0.1053	-0.0863
10/500/2	-0.0756	-0.1776	-0.1168	-0.0882
10/1000/2	-0.0928	-0.1839	-0.0993	-0.1084
Overall	-0.1064	-0.1533	-0.1053	-0.1086

*Values in the table are the averages of the results of five different datasets for each problem setting.

Table 5.13 shows the p values of paired t tests for AC, CC and WC values between each ensemble and ECDST. At 5% significance level, ECDST is more effective in raising the probability of belief degree of true propositions (AC) and correctly predicted targets (CC) than ECY, ECMDF and ECMDFS. Also, it lowers the average probability change in wrong predictions (WC) less than ECY. There is not statistically significant difference between ECMDF and ECDST in WC.

Table 5.13. *p values of hypothesis testing for AC, CC and WC values between each ensemble and ECDST*

Performance Measure	ECY	ECMDF	ECMDFS
AC	0.0014	0.0010	0.0005
CC	0.0026	0.0011	0.0003
WC	0.0003	0.8631	0.6263

The p values of hypothesis testing for AC, CC and WC values between each ensemble and ECY are given in Table 5.14. ECDST and ECMDFS give higher probabilities in AC and CC than ECY. Also, ECMDF give better performances for AC. However, ECY and ECMDF give similar results in CC. ECY also decreases the true propositions probability of belief degree (WC) more than the other ensembles.

Table 5.14. *p values of hypothesis testing for AC, CC and WC values between each ensemble and ECY*

Performance Measure	ECDST	ECMDF	ECMDFS
AC	0.0014	0.0111	0.0025
CC	0.0026	0.7733	0.0053
WC	0.0003	0	0.0005

[1,2], [1,5], [1,10], [1,20], [1,40] are used as scaling intervals to represents the full spectrum of the L_p metric. We record the number of times of each scaling interval used and report them in Table 5.15. Table 5.15 shows that the adaptive distance idea works in representing the distance to achieve higher PCP values.

Table 5.15. *Number of times of each scaling interval used for ECMDF and ECMDFS*

Scaling Interval	**ECMDF**	**ECMDFS**
[1, 2]	55	61
[1, 5]	10	4
[1, 10]	6	5
[1, 20]	8	7
[1, 40]	11	13

We also record the average accuracy ratios of ANN and SVM for each sensor in each problem setting and give them in Table 5.16. We observe that ANN gives better classification accuracies than SVM for noisy sensors.

Table 5.16. *Average classification accuracies of ANN and SVM for each problem setting*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	0.78	0.76	0.74	0.91	0.86	0.84
5/500/0	0.84	0.78	0.76	0.89	0.82	0.86
5/1000/0	0.84	0.77	0.82	0.87	0.83	0.86
5/100/1	0.78	0.76	0.38	0.91	0.86	0.35
5/500/1	0.84	0.78	0.37	0.89	0.82	0.33
5/1000/1	0.84	0.77	0.36	0.87	0.83	0.33
5/100/2	0.78	0.39	0.38	0.91	0.34	0.35
5/500/2	0.84	0.36	0.37	0.89	0.34	0.33
5/1000/2	0.84	0.35	0.36	0.87	0.33	0.33
10/100/0	0.84	0.83	0.81	0.91	0.94	0.90
10/500/0	0.85	0.78	0.82	0.95	0.87	0.89
10/1000/0	0.87	0.79	0.82	0.95	0.87	0.92
10/100/1	0.84	0.83	0.50	0.91	0.94	0.30
10/500/1	0.85	0.78	0.43	0.95	0.87	0.33
10/1000/1	0.87	0.79	0.39	0.95	0.87	0.33
10/100/2	0.84	0.52	0.50	0.91	0.31	0.30
10/500/2	0.85	0.39	0.43	0.95	0.34	0.33
10/1000/2	0.87	0.39	0.39	0.95	0.32	0.33
Overall	0.84	0.66	0.53	0.91	0.69	0.51

*Values in the table are the averages of the results of five different datasets for each problem setting.

Table 5.17 shows the computational times. As the number of noisy sensors increases, computational times for all of the ensemble increases. However, all models have approximately similar computational times.

Table 5.17. *Computational times of each ensemble*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	107	4	139	6	149	10	212	4
5/500/0	201	8	192	12	236	8	243	6
5/1000/0	233	13	223	17	300	11	311	10
5/100/1	192	8	181	9	195	7	203	6
5/500/1	436	10	424	12	473	9	480	7
5/1000/1	933	15	915	19	1003	13	1016	12
5/100/2	195	7	186	8	201	7	207	5
5/500/2	647	10	634	12	689	8	688	8
5/1000/2	1407	12	1394	17	1317	13	1484	11
10/100/0	195	8	181	9	197	8	206	6
10/500/0	218	10	206	13	256	9	265	8
10/1000/0	307	16	292	20	380	13	394	12
10/100/1	195	8	180	9	196	8	206	5
10/500/1	500	11	485	14	539	9	547	7
10/1000/1	1516	13	1499	17	1585	12	1595	10
10/100/2	169	6	184	8	197	7	205	5
10/500/2	763	12	750	12	796	9	809	9
10/1000/2	3201	14	3193	17	3268	12	3281	12
Overall	634	10	625	13	665	10	686	8

*Values in the table are the averages of the results of five different datasets for each problem setting.

5.5. Dataset Generation for Paradoxical Situations

DST is an effective data fusion algorithm to combine separate pieces of evidences coming from different sensors. However, it gives unreasonable results under certain

paradoxical situations. Our main purpose in this study is to propose a new method that successfully handles paradoxical situations through elastic distance calculation in credibility degrees. However, datasets generated for each problem setting in Table 5.1 do not contain any paradoxical situations, which occur under certain combination of evidences for the same target. By only changing the parameters in problem settings, we are not able to directly control the probabilistic classification results of ANN and SVM to generate those paradoxes. Thus, we manipulate the probabilistic classification results of classifiers for both training and test datasets in all ensembles in the same manner to generate the paradoxical situations and test them.

The following example is used as an example to show the dataset generation for all paradoxical situations.

For example; Suppose that there are three different propositions in FOD as A, B, C.

$FOD = \{A, B, C\}$

Suppose also that there are 3 different evidences for the same target m and the class of the target is C.

$mt_{m1} = \{0.02, 0.03, 0.95\}$

$mt_{m2} = \{0.04, 0.78, 0.18\}$

$mt_{m3} = \{0.7, 0.1, 0.2\}$

Where mt_{mi} is the piece of evidence of sensor i for target m for test dataset.

Below, we provide details of generating paradoxical datasets.

Complete Conflict Paradox

Complete Conflict Paradox occurs when conflicting degree is equal to 1. In this paradox, classical DST cannot be applied as the denominator of the combination formula gets 0, i.e. Equation (2.20).

In order to make conflicting degree equal to 1, we manipulate the example evidences as follows:

$mt_{m1} = \{0,0,1\}$

$mt_{m2} = \{1,0,0\}$

$mt_{m3} = \{0.7,0.1,0.2\}$

For the first sensor, we change the true class' BPA as 1. For the second sensor, if the minimum BPA in FOD belongs to true class, we change the maximum BPA in FOD as 1 and change the other BPAs as 0. On the other hand, if the minimum BPA in FOD does not belong to true class, we change the minimum BPA as 1 and change the other BPAs as 0.

0 Trust Paradox

In 0 Trust Paradox, if one of the sensors totally denies one of the classes, then resulting combined mass of it always gets 0.

In order to generate this paradox, we change one of the sensors' the true class' BPA as 0. If true class' BPA occurs at the minimum BPA of FOD, we add its BPA to maximum BPA. Otherwise, we add true class' BPA to minimum BPA in FOD.

In order to generate 0 Trust Paradox, we manipulate the example evidences as follows:

$mt_{m1} = \{0.97, 0.03,0\}$

$mt_{m2} = \{0.04,0.78,0.18\}$

$mt_{m3} = \{0.7,0.1,0.2\}$

1 Trust Paradox

This paradox is an extension of 0 Trust Paradox. In order to generate this paradox, we manipulate the 2 propositions' BPA for 2 different sensors. Manipulated evidences for the example evidences are as follows:

$$mt_{m1} = \{0.97, 0.03, 0\}$$

$$mt_{m2} = \{0, 0.82, 0.18\}$$

$$mt_{m3} = \{0.7, 0.1, 0.2\}$$

For example, if the true class of the target is C, we change the BPA of it in the first sensor as 0 and add its value to the first proposition's BPA. In addition, we change the BPA of A in the second sensor's evidence as 0 and add its value to second proposition's BPA.

High Conflict Paradox

Under high conflict, DST may give unsatisfactory results. To generate High Conflict Paradox, we firstly apply the algorithm to generate Complete Conflict Paradox. Then, we add very small mass to BPAs whose values are 0. Lastly, we subtract the added masses from BPA that has the maximum mass for each evidence and target.

Manipulated version of the example evidences are as follows:

$$mt_{m1} = \{0.002, 0.003, 0.995\}$$

$$mt_{m2} = \{0.004, 0.978, 0.018\}$$

$$mt_{m3} = \{0.7, 0.1, 0.2\}$$

5.6. Computational Results for Paradoxical Situations

For each of the paradoxical situations, five different datasets with 3 different sensors and number of classes as 3 where each of them has a size of 1000 samples are generated and averages of PCP values are presented in Table 5.18.

In Complete Conflict Paradox, DST cannot be applied as conflict degree is equal to 1 and thus, denominator in the combination rule gets 0. On the other hand, in 0 Trust Paradox, we change the BPA of true class as 0 in one of the sensors for each target. Thus, DST give 0 mass to this class. The same situation is also valid for 1 Trust Paradox. As a results, DST gives 0% PCP values for both of the paradoxes

For all paradoxes, ECMDFS outperforms the others. In addition, ECMDF gives similar results with ECMDFS. ECY's performance under the paradoxes is satisfactory but not good as proposed algorithms.

Table 5.18. *Average PCP values for paradoxical situations*

Paradox	**ECDST (%)**	**ECY (%)**	**ECMDF (%)**	**ECMDFS (%)**
Complete Conflict Paradox	Not Applicable	91.94	98.06	98.42
0 Trust Paradox	0.00	73.45	81.94	82.85
1 Trust Paradox	0.00	77.82	82.79	83.82
High Conflict Paradox	87.21	91.94	98.06	98.30

*Values in the table are the averages of the results of five different datasets for each paradox.

CHAPTER 6

HYBRID MULTI-SENSOR TARGET CLASSIFICATION MODELS

In this chapter, we present the hybrid extensions of both ECMDF and ECMDFS. In Section 6.1, details of the hybrid models are given. In Section 6.2, parameter settings are given. Computational results are presented in Section 6.3.

6.1. The Hybrid Models

While the classical DST combination rule gives unsatisfactory results under paradoxical situations, the proposed algorithms handle those cases well. Thus, we combine the power of both the classical DST and the proposed algorithms, and propose the hybrid extensions of both ECMDF and ECMDFS. We abbreviate the hybrid ECMDF as h-ECMDF, and the hybrid ECMDFS as h-ECMDFS.

In the hybrid models, the main idea is to use the classical DST when the conflicting degree is low, and to use the proposed approaches otherwise. When the conflicting degree is lower than the threshold value, the hybrid model also checks if any evidence coming from multiple sensors for the respective target contains BPA whose value is 0. The latter condition ensures that if paradoxical situations except the *high conflict paradox* is contained in the dataset, then ECMDF/ECMDFS is employed to fuse those evidences. One can see from Section 5.5 that if data contains any paradox except the *high conflict paradox*, one of the BPAs is equal to 0. By considering both of the conditions in the hybrid models, we ensure the model to use ECMDF/ECMDFS when there is any paradox in dataset, and to use classical DST otherwise.

h-ECMDF and h-ECMDFS are inspired from Zhang et al. (2017) in activation of the proposed approaches. Like Zhang et al. (2017), we control the conflicting degree

through a threshold value, ξ. If the conflicting degree is smaller than ξ, then the evidences are fused by the classical DST. If the conflicting degree is larger than ξ, then the evidences are fused by the proposed approaches. Figure 6.1 shows the flowchart of the hybrid models.

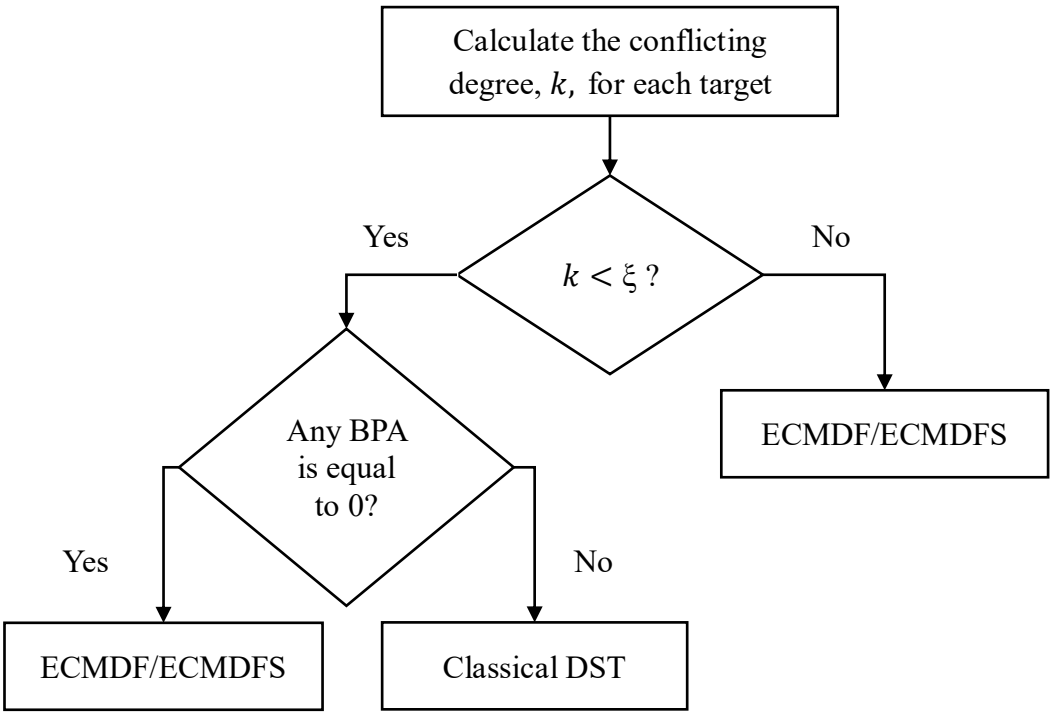

Figure 6.1. Flowchart of hybrid models

6.2. Parameter Settings

When developing these approaches, we need to find a way to define a reasonable threshold value that determines which model to use. From Table 5.7 in Section 5.4, we can see that the largest positive difference between PCP values of proposed algoritms and ECDST occurs in problem setting 10/100/1 with 1.82% (73.94%-72.12%) with ECMDFS. Also, the largest negative difference occurs in problem setting 10/100/2 with -7.88% (61.82%-69.70%) for ECMDFS. By using the hybrid

models, we aim to increase the PCP values. Thus, to determine a threshold value of both of the hybrid models, we solve them with different levels of conflicting degree, k, $k \in \{0.75, 0.80, 0.85, 0.90, 0.95, 1\}$, for problem setting 10/100/1 and 10/100/2 with 5 different datasets for each problem setting. For each problem setting, we observe different PCP values in only one of the seeds. For those seeds, Figures 6.2 and 6.3 show the number of correctly predicted targets versus different values of k for problem settings 10/100/1 and 10/100/2, respectively. Tables 6.1 and 6.2 show the number of activation times of each algorithm. Detailed results of the computational study of different values of conflicting degree are given in Appendix G.

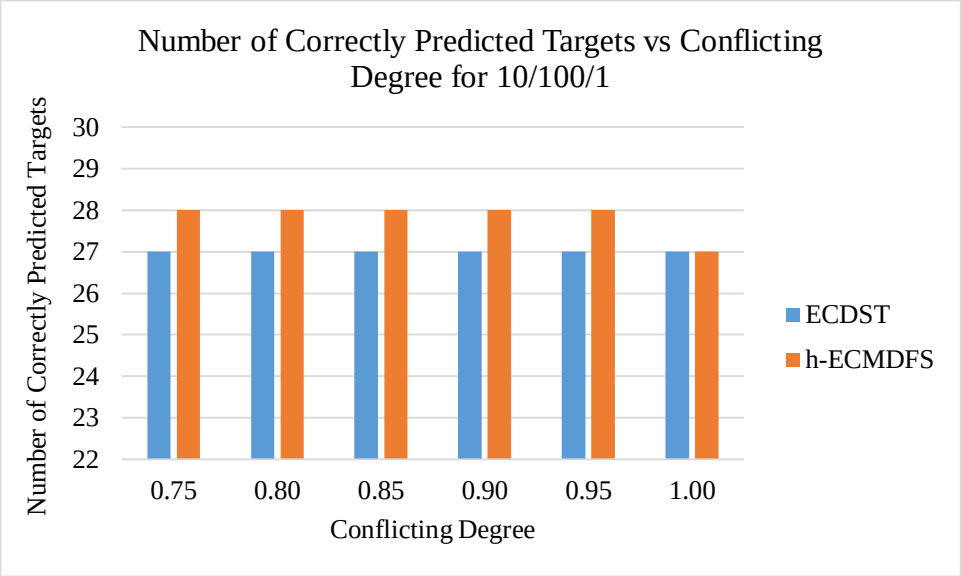

Figure 6.2. Number of correctly predicted targets versus different k values for problem setting 10/100/1

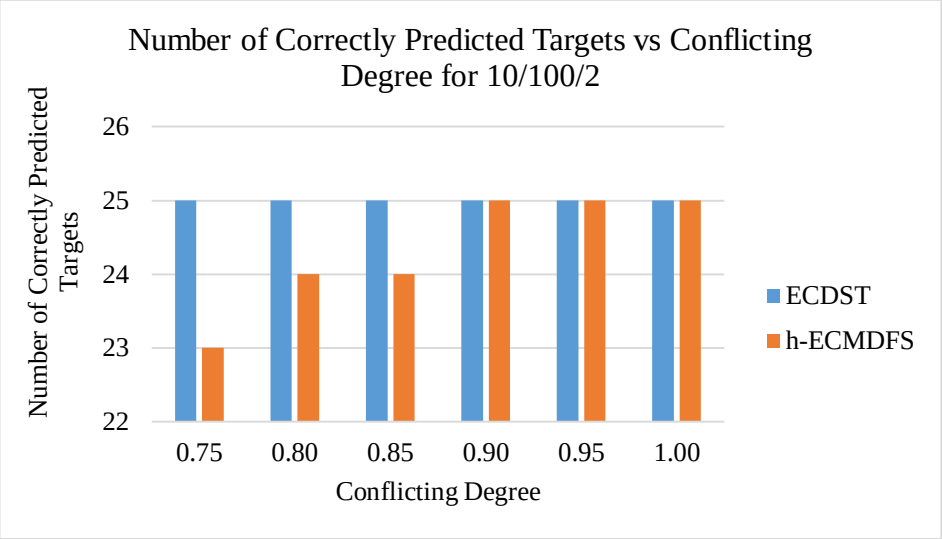

Figure 6.3. Number of correctly predicted targets versus different k values for problem setting 10/100/2

Table 6.1. *Number of activation times of each algorithm for problem setting 10/100/1*

Conflicting Degree	**Number of Activation Times**	
	ECDST	**ECMDFS**
0.75	29	4
0.80	29	4
0.85	30	3
0.90	30	3
0.95	31	2
1	33	0

Table 6.2. *Number of activation times of each algorithm for problem setting 10/100/2*

Conflicting Degree	**Number of Activation Times**	
	ECDST	**ECMDFS**
0.75	29	4
0.80	31	2
0.85	32	1
0.90	33	0
0.95	33	0
1	33	0

In problem setting 10/100/1, ECMDFS gives better PCP value than ECDST. In Figure 6.2, we can see that if we use h-ECMDFS with different threshold values ranging from 0.75 to 1, we get the same PCP values for all conflicting degrees except for 1. Also, when the threshold value is equal to 1, h-ECMDFS becomes the same with ECDST. In this situation, all evidences are combined with the classical DST.

In Figure 6.3, increasing the threshold value also increases the PCP values of h-ECMDFS. When the conflicting degree is increased to 0.8, h-ECMDFS predicts one more target as true. This situation also valid, when k = 0.85. Additional improvements are observed when the threshold value is increased to 0.90 or higher. However, for conflicting degrees greater than 0.85, the hybrid algorithm uses the classical DST for combining evidences. By using hybrid models, we are to combine the power of both algorithms. Thus, both ECMDFS and classical DST should be used to benefit the strengths of both of them. A careful investigation on the generated data shows that, if the threshold value is set to 0.90 or higher, ECMDFS is not activated. By considering the number of activation times of ECMDFS in both of the problem settings, 0.80 is chosen as threshold value, ξ, for experimental purposes.

Note that the number of paradoxical situations in generated data is small. Thus, we may expect that the marginal improvements in performances of the hybrid models will be limited.

6.3. Computational Results for Hybrid Models

h-ECMDF and h-ECMDFS are solved for 18 different problem settings in Table 5.1. In each problem settings, the same datasets to test ECMDF and ECMDFS are used, and average PCP results for each problem setting for ECDST, ECMDF, ECMDFS, h-ECMDF and h-ECMDFS are presented in Table 6.1. The detailed experimental results are presented in Appendix H. Note that the results of ECDST, ECMDF and ECMDFS in Table 6.1 are the same as in Table 5.7.

The last line of Table 6.3 shows the overall averages of all problem settings for each ensemble. By the hybrid extension of the proposed algorithms, we get 6.41% (82.22%-75.81) and 0.31% (82.86%-82.55%) increase in overall average PCP values for ECMDF and ECMDFS, respectively.

Table 6.3. *PCP values of the hybrid models*

Problem Setting	**PCP (%)**				
	ECDST	**ECMDF**	**ECMDFS**	**h-ECMDF**	**h-ECMDFS**
5/100/0	81.82	81.82	81.82	81.21	81.21
5/500/0	91.15	90.30	90.42	91.03	90.79
5/1000/0	88.73	88.24	88.24	88.67	88.73
5/100/1	80.61	81.21	81.82	80.61	80.61
5/500/1	89.45	87.88	88.85	88.12	88.73
5/1000/1	86.91	85.03	85.76	85.94	86.30
5/100/2	72.73	43.64	73.33	68.48	73.33
5/500/2	85.21	64.97	85.82	84.85	85.21
5/1000/2	82.42	79.03	82.61	82.42	82.42
10/100/0	77.58	77.58	77.58	76.36	76.97
10/500/0	88.97	88.24	88.73	88.61	88.85
10/1000/0	89.09	88.42	88.30	88.61	88.61
10/100/1	72.12	72.12	73.94	72.12	72.73
10/500/1	87.39	82.42	86.42	86.91	87.52
10/1000/1	86.36	83.09	85.21	84.61	85.45
10/100/2	69.70	49.70	61.82	68.48	69.09
10/500/2	81.21	57.33	80.97	79.76	81.09
10/1000/2	83.82	63.58	84.30	83.21	83.82
Overall	83.07	75.81	82.55	82.22	82.86

*Values in the table are the averages of the results of five different datasets for each problem setting.

Out of 18 problem settings, h-ECMDF gives better PCP values for 14 settings than ECMDF. Also, h-ECMDFS's PCP values are higher than ECMDFS's for 9 settings. We can see that the largest performance increase by the hybrid extensions occurs in the problem settings with 2 noisy sensors for ECMDF. The largest difference between ECMDF and ECDST, which occurs in the problem setting 5/100/2. (i.e. a performance

difference of -29.09% decreases to -4.24% (68.48%-72.73%)). The largest difference for ECMDFS decreases from -7.88% to -0.61% (69.09%-69.70%) in the problem setting 10/100/2. However, the maximum positive difference between ECMDFS and ECDST decreases from 1.82% to 0.61% (72.73%-72.12%).

Table 6.4 shows the activation times of the classical DST and ECMDF/ECMDFS in each problem setting. In general, when the number of samples increases, activation time of ECMDF/ECMDFS increases. However, the majority of the evidences for each target in each problem setting is fused with the classical DST. That's why the average PCP values in Table 6.3 are similar to each other for all ensembles.

Table 6.4. *Summary of activation times of hybrid models and the classical DST in each problem setting*

Problem Setting	**Number of DST Used**			**Number of ECMDF/ECMDFS Used**		
	Min	**Average**	**Max**	**Min**	**Average**	**Max**
5/100/0	31	32.40	33	0	0.60	2
5/500/0	152	157.80	162	3	7.20	13
5/1000/0	303	315.00	327	3	15.00	27
5/100/1	31	32.40	33	0	0.60	2
5/500/1	152	157.80	162	3	7.20	13
5/1000/1	303	315.00	327	3	15.00	27
5/100/2	26	30.40	33	0	2.60	7
5/500/2	161	164.20	165	0	0.80	4
5/1000/2	330	330.00	330	0	0.00	0
10/100/0	29	32.00	33	0	1.00	4
10/500/0	158	161.00	163	2	4.00	7
10/1000/0	304	309.80	316	14	20.20	26
10/100/1	29	32.00	33	0	1.00	4
10/500/1	158	161.00	163	2	4.00	7
10/1000/1	304	309.80	316	14	20.20	26
10/100/2	26	30.80	33	0	2.20	7
10/500/2	151	161.60	165	0	3.40	14
10/1000/2	321	327.00	330	0	3.00	9

*Values in the table are the averages of the results of five different datasets for each problem setting.

We performed paired t-test to see the statistical significance. Table 6.5 shows p values of hypothesis testing of PCP values of h-ECMDF, h-ECMDFS and ECDST. At 5% significance level, ECDST and h-ECMDFS give similar results. However, at the same significance level, ECDST's performance is better than h-ECMDF.

Table 6.5. *p values of hypothesis testing for PCP values between h-ECMDF, h-ECMDFS and ECDST*

Ensemble	**h-ECMDF**	**h-ECMDFS**
ECDST	0.0024	0.0529

We also performed paired t-test between proposed algorithms and their hybrid extensions. At 5% significance level, we improved the performance of ECMDF by h-ECMDF statistically. However, there is not statistically difference between the performances of ECMDFS and h-ECMDFS.

In Table 6.6, AC, CC and WC values for h-ECMDF and h-ECMDFS are given. Both h-ECMDF and h-ECMDFS are successful in increasing AC and CC. Yet, they decrease WC for all problem settings. Contrary to the behavior of ECMDF in problem settings when the number of noisy sensors is 2, i.e. problem settings 5/100/2, 5/500/2, 5/1000/2, 10/100/2, 10/500/2 and 10/1000/2, h-ECMDF give much better performances. That is h-ECMDF successfully increases CC for all problem settings.

Table 6.6. *AC, CC and WC values for hybrid models*

Problem Setting	h-ECMDF			h-ECMDFS		
	AC	**CC**	**WC**	**AC**	**CC**	**WC**
5/100/0	0.1672	0.2370	-0.1333	0.1674	0.2374	-0.1334
5/500/0	0.1456	0.1740	-0.1418	0.1470	0.1752	-0.1253
5/1000/0	0.1475	0.1857	-0.1391	0.1479	0.1863	-0.1445
5/100/1	0.2181	0.2921	-0.0782	0.2180	0.2920	-0.0782
5/500/1	0.2455	0.2931	-0.1004	0.2493	0.2941	-0.0927
5/1000/1	0.2447	0.3032	-0.1137	0.2478	0.3046	-0.1113
5/100/2	0.1534	0.2781	-0.1141	0.1859	0.2702	-0.0382
5/500/2	0.2837	0.3413	-0.0780	0.2850	0.3409	-0.0766
5/1000/2	0.2729	0.3442	-0.0657	0.2729	0.3442	-0.0657
10/100/0	0.1434	0.2235	-0.1083	0.1439	0.2226	-0.1138
10/500/0	0.1683	0.2076	-0.1392	0.1687	0.2075	-0.1424
10/1000/0	0.1397	0.1741	-0.1286	0.1396	0.1740	-0.1300
10/100/1	0.1534	0.2482	-0.0993	0.1537	0.2461	-0.1013
10/500/1	0.2514	0.3064	-0.1121	0.2530	0.3059	-0.1181
10/1000/1	0.2217	0.2854	-0.1252	0.2264	0.2847	-0.1162
10/100/2	0.1749	0.2862	-0.0863	0.1764	0.2860	-0.0966
10/500/2	0.2403	0.3217	-0.0916	0.2475	0.3192	-0.0740
10/1000/2	0.2776	0.3541	-0.1020	0.2808	0.3532	-0.0942
Overall	0.2027	0.2698	-0.1087	0.2062	0.2691	-0.1029

* Values in the table are the averages of the results of five different datasets for each problem setting.

Performance of ECMDF and ECMDFS under paradoxical situations was investigated in Section 5.5 and 5.6. Both of the algorithms outperformed the classical DST and the modified DST proposed by Yong et al. (2004). Thus, we do not investigate the performance of h-ECMDF and h-ECMDFS under paradoxical situations further.

CHAPTER 7

CONCLUSION

In this thesis, we address the classification phase of ATR systems. For the classification purpose, we employ ensemble of classifiers and focus on non-imaginary multiple heterogeneous data sources. In the ensemble, both ANN and SVM are trained for each sensor and the classifier that has the highest accuracy ratio is chosen. Then, different results coming from each classifier for each sensor are combined through modified DST.

Our main objective in this study is to propose a new method that elastically calculates the distances between each evidence and modify them to overcome the paradoxical situations in DST, which are Complete Conflict Paradox, 0 Trust Paradox, 1 Trust Paradox and High Conflict Paradox. In all proposed methods, L_p distance metric is used. This distance metric enables us to calculate the distances between evidences by taking into account the effects of each sensor. Sensor accuracies calculated in training phase are used as weights in the distance metric. Also, pairwise correlation coefficients between sensors calculated in the training phase is used as p parameter of L_p distance metric.

Two different methods are proposed which are Ensemble of Classifiers with Modified Distance Function (ECMDF) and Ensemble of Classifiers with Modified Distance Function and Sensor Accuracy (ECMDFS). The first one discounts the evidences by credibility degrees whereas the second one discounts based on both credibility degrees and sensor accuracies. Both of the models are inspired from Yong et al. (2004) in discounting the evidences. They calculate the distances for each evidence pairs for the same target. They use Jousselme Distance function. However, ECMDF and ECMDFS

integrate DST with ML and MCDM, and calculate credibility degrees and sensor accuracies based on the training datasets.

Computational results of ECMDF and ECMDFS are compared with the ensembles that integrates the classical DST and the modified DST proposed by Yong et al. (2004) through the generated datasets that represent artificial ATR environment. We use 18 different problem settings in computational results, and generate 5 different datasets for each problem setting. The results shows that ECY and ECMDF perform poorly when the number of noisy sensors is more than the consistent ones. In this situation, credibility degrees between noisy sensors are high and thus, the effect of the evidences coming from them are high in the evidence combination phase. The results also show that there is not statistically significant difference between ECDST and ECDMFS in the computational experiments. In addition, ECMDFS shows better performance in problem settings when the number of features is low and number of noisy sensors is at maximum. Moreover, ECDST and ECMDFS outperform ECY.

The behavior of all ensembles in the paradoxical situations is also investigated. Probabilistic classification results of each classifier for each sensor are manipulated and artificial datasets are generated. The results show that ECMDF and ECMDFS give the best performances in all paradoxes.

We also present the hybrid extensions of ECMDF and ECMDFS as h-ECMDF and h-ECMDFS. In the proposed algorithms, if the conflicting degree is smaller than the threshold value, classical DST is activated. Otherwise, ECMDF or ECMDFS is activated. Both hybrid models are tested with the same problem settings and datasets with ECMDF and ECMDFS. By the hybrid extension of ECMDF, we are able to get better performance measures for the majority of the problem settings. Also, by hybrid extension of ECMDFS, we make ECMDFS to give satisfactory results for all problem settings. By h-ECMDFS, we improve the worst performance of ECMDFS.

To the best of our knowledge, there is not any study that consider adaptive distance metric in Dempster-Shafer Theory for assigning the reliability of the sources in the

literature. The main contribution of the study is to integrate classical DST with adaptive distance metric, L_p metric, for assigning the credibility degrees of the each sensor. The proposed distance metric calculates the distances scenario based and flexible to the different types of datasets. Also, the study combines DST with ML and MCDM. The proposed way of combination of different pieces of evidences through L_p metric by modified DST is proven to be effective in both computational results and paradoxical situations.

ECMDFS and h-ECMDFS are proven to be more successful than ECMDF and h-ECMDF. ECMDFS and h-ECMDFS give statistically similar results with the classical DST, and the proposed way of discounting the evidences by both credibility degrees and sensor accuracies outperforms the classical DST in all kinds of paradoxes. Although ECMDF also gives more satisfactory results in paradoxes than the classical DST does, classical DST gives better performances for non paradoxical datasets than ECMDF and h-ECMDF.

All ensembles in this study are tested with generated artificial datasets. In real life applications, finding multi-sensor datasets in sensor data fusion is problematic. A future direction would be testing the proposed algorithms with real datasets.

Sensors may have different detection capabilities against different types of targets. For example, one type of sensor may have larger detection capability if the target is a bomber aircraft, and have smaller detection capability if the target is stealth aircraft. Another future direction would be assigning target specific credibility degrees.

REFERENCES

Ali, K. (1995). On the Link between Error Correlation and Error Reduction in Decision Tree Ensembles. doi:10.1.1.51.1594

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32.

Camargo, L. S., & Yoneyama, T. (2001). Specification of Training Sets and the Number of Hidden Neurons for Multilayer Perceptrons. *Neural Computation, 13*(12), 2673-2680. doi:10.1162/089976601317098484

Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing, 275*, 278-287. doi:10.1016/j.neucom.2017.08.040

Chen, J., Ye, F., Jiang, T., & Tian, Y. (2017). Conflicting Information Fusion Based on an Improved DS Combination Method. *Symmetry, 9*(11), 278. doi:10.3390/sym9110278

Chen, L., Diao, L., & Sang, J. (2018). Weighted Evidence Combination Rule Based on Evidence Distance and Uncertainty Measure: An Application in Fault Diagnosis. *Mathematical Problems in Engineering, 2018*, 1-10. doi:10.1155/2018/5858272

Chen, Y., Cremers, A. B., & Cao, Z. (2014). Interactive color image segmentation via iterative evidential labeling. *Information Fusion, 20*, 292-304. doi:10.1016/j.inffus.2014.03.007

Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications.

Dempster, A. P. (1967). Upper and Lower Probabilities Induced by a Multivalued Mapping. *The Annals of Mathematical Statistics, 38*(2), 325-339. doi:10.1214/aoms/1177698950

Deng, Y., & Chan, F. T. (2011). A new fuzzy dempster MCDM method and its application in supplier selection. *Expert Systems with Applications, 38*(8), 9854-9861. doi:10.1016/j.eswa.2011.02.017

Dong, G., & Kuang, G. (2015). Target Recognition via Information Aggregation Through Dempster–Shafers Evidence Theory. *IEEE Geoscience and Remote Sensing Letters, 12*(6), 1247-1251. doi:10.1109/lgrs.2015.2390914

Dubois, D., & Prade, H. (1988). Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence, 4*(3), 244-264. doi:10.1111/j.1467-8640.1988.tb00279.x

Dymova, L., & Sevastjanov, P. (2010). An interpretation of intuitionistic fuzzy sets in terms of evidence theory: Decision making aspect. *Knowledge-Based Systems, 23*(8), 772-782. doi:10.1016/j.knosys.2010.04.014

Fan, C., Song, Y., Lei, L., Wang, X., & Bai, S. (2018). Evidence reasoning for temporal uncertain information based on relative reliability evaluation. *Expert Systems with Applications, 113*, 264-276. doi:10.1016/j.eswa.2018.06.048

Fan, X., & Zuo, M. J. (2006). Fault diagnosis of machines based on D–S evidence theory. Part 2: Application of the improved D–S evidence theory in gearbox fault diagnosis. *Pattern Recognition Letters, 27*(5), 377-385. doi:10.1016/j.patrec.2005.08.024

Florea, M. C., Jousselme, A., Bossé, É, & Grenier, D. (2009). Robust combination rules for evidence theory. *Information Fusion, 10*(2), 183-197. doi:10.1016/j.inffus.2008.08.007

Foucher, S., Germain, M., Boucher, J., & Benie, G. (2002). Multisource classification using ICM and Dempster-Shafer theory. *IEEE Transactions on Instrumentation and Measurement, 51*(2), 277-281. doi:10.1109/19.997824

Freund, Y. (1995). Boosting a Weak Learning Algorithm by Majority. *Information and Computation, 121*(2), 256-285. doi:10.1006/inco.1995.1136

Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning, 41*(3), 315-343.

Genton, M. G. (2001). Classes of Kernels for Machine Learning: A Statistics Perspective. *Journal of Machine Learning Research,2*, 299-312. doi:10.1162/15324430260185646.

Géron, A. (2018). *Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligent systems*. Beijing, OReilly.

Guyon, I. (2003). Design of experiments for the NIPS 2003 variable selection benchmark.

Horiuchi, T. (1998). Decision rule for pattern classification by integrating interval feature values. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(4), 440-448. doi:10.1109/34.677286

Inagaki, T. (1991). Interdependence between safety-control policy and multiple-sensor schemes via Dempster-Shafer theory. *IEEE Transactions on Reliability, 40*(2), 182-188. doi:10.1109/24.87125

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning: With applications in R*. New York: Springer.

Jonathan Milgram, J., Cheriet, M., Sabourin, R. (2006) "One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?. *Tenth International Workshop on Frontiers in Handwriting Recognition*, Université de Rennes 1, Oct 2006, La Baule (France).

Josang, A., Daniel, M., & Vannoorenberghe, P. (2003). Strategies for combining conflicting dogmatic beliefs. *Sixth International Conference of Information Fusion, 2003. Proceedings of the*. doi:10.1109/icif.2003.177365

Jurek, A., Bi, Y., Wu, S., & Nugent, C. (2013). A survey of commonly used ensemble-based classification techniques. *The Knowledge Engineering Review, 29*(5), 551-581. doi:10.1017/s0269888913000155

Kavzoglu, T., & Colkesen, I. (2009). A kernel functions analysis for support vector machines for land cover classification. *International Journal of Applied Earth Observation and Geoinformation, 11*(5), 352-359. doi:10.1016/j.jag.2009.06.002

Ke, J., & Liu, X. (2008). Empirical Analysis of Optimal Hidden Neurons in Neural Network Modeling for Stock Prediction. *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2*, 828-832. doi:10.1109/paciia.2008.363

Keerthi, S. S., & Gilbert, E. G. (2002). Convergence of a Generalized SMO Algorithm for SVM Classifier Design. *Machine Learning, 46*(1-3), 351-360.

Khaleghi, B., Khamis, A., Karray, F. O., & Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion, 14*(1), 28-44. doi:10.1016/j.inffus.2011.08.001

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica, 31,* 249-268.

Kulkarni, V., & Sinha, P. (2013). Random forest classifiers: A survey and future research directions. *International Journal of Advanced Computing,* 1144-1153.

Ladjal, M., Bouamar, M., Djerioui, M., & Brik, Y. (2016). Performance evaluation of ANN and SVM multiclass models for intelligent water quality classification using Dempster-Shafer Theory. *2016 International Conference on Electrical and Information Technologies (ICEIT)*. doi:10.1109/eitech.2016.7519588

Leung, Y., Ji, N., & Ma, J. (2013). An integrated information fusion approach based on the theory of evidence and group decision-making. *Information Fusion, 14*(4), 410-422. doi:10.1016/j.inffus.2012.08.002

Li, W., Leung, H., Kwan, C., & Linnell, B. (2008). E-Nose Vapor Identification Based on Dempster–Shafer Fusion of Multiple Classifiers. *IEEE Transactions on Instrumentation and Measurement, 57*(10), 2273-2282. doi:10.1109/tim.2008.922092

Li, Y., Chen, J., Ye, F., & Liu, D. (2016). The Improvement of DS Evidence Theory and Its Application in IR/MMW Target Recognition. *Journal of Sensors, 2016*, 1-15. doi:10.1155/2016/1903792

Liu, Z., Pan, Q., Dezert, J., & Martin, A. (2018). Combination of Classifiers With Optimal Weight Based on Evidential Reasoning. *IEEE Transactions on Fuzzy Systems, 26*(3), 1217-1230. doi:10.1109/tfuzz.2017.2718483

Mcculloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115-133. doi:10.1007/bf02478259

Mercier, D., Quost, B., & Denœux, T. (2008). Refined modeling of sensor reliability in the belief function framework using contextual discounting. *Information Fusion, 9*(2), 246-258. doi:10.1016/j.inffus.2006.08.001

Moore, G. E. (2006). Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter, 11*(3), 33-35. doi:10.1109/n-ssc.2006.4785860

Murphy, C. K. (2000). Combining belief functions when evidence conflicts. *Decision Support Systems, 29*(1), 1-9. doi:10.1016/s0167-9236(99)00084-6

Pal, N., & Ghosh, S. (2001). Some classification algorithms integrating Dempster-Shafer theory of evidence with the rank nearest neighbor rules. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 31*(1), 59-66. doi:10.1109/3468.903867

Platt, J. C. (1999). Using Analytic QP and Sparseness to Speed Training of Support Vector Machines. In *In Neural Information Processing Systems 11* (pp. 557-563). MIT Press.

Rogers, S. K., Colombi, J. M., Martin, C. E., Gainey, J. C., Fielding, K. H., Burns, T. J., Oxley, M. U. (1995). Neural networks for automatic target recognition. *Neural Networks, 8*(7-8), 1153-1184. doi:10.1016/0893-6080(95)00050-X

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536. doi:10.1038/323533a0

Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8*(4), 1-18. doi:10.1002/widm.1249

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117. doi:10.1016/j.neunet.2014.09.003

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton: Princeton University Press.

Sheela, K. G., & Deepa, S. N. (2013). Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering, 2013*, 1-11. doi:10.1155/2013/425740

Siddique, M., & Tokhi, M. (2001). Training neural networks: Backpropagation vs. genetic algorithms. *IJCNN01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, 2673-2678. doi:10.1109/ijcnn.2001.938792

Smarandache, F., & Dezert, J. (2006). *Advances and applications of DSmT for information fusion: Collected works*. Rehoboth, NM: American Research Press.

Smets, P. (1990). The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(5), 447-458. doi:10.1109/34.55104

Smets, P., & Kennes, R. (1994). The transferable belief model. *Artificial Intelligence, 66*(2), 191-234.

Tian, N., Fleurant, A., Kuimova, A., Wezeman, P., & Wezeman, S. (2017). Military spending in 2017. Retrieved from http://visuals.sipri.org/

Vapnik, V., & Chervonenkis, A. (1964). A note on one class of perceptrons. *Automation and Remote Control, 25*.

Vapnik, V., & Cortes, C. (1995). Support-Vector Networks. *Machine Learning, 20*(3), 273-297.

Vivarelli, F., & Williams, C. K. (2001). Comparing Bayesian neural network algorithms for classifying segmented outdoor images. *Neural Networks, 14*(4-5), 427-437. doi:10.1016/s0893-6080(01)00024-7

Voorbraak, F. (1991). On the justification of Dempsters rule of combination. *Artificial Intelligence, 48*(2), 171-197. doi:10.1016/0004-3702(91)90060-w

Wei-Wei, W., & Li-Na, B. (2012). Target Identification Based on Neural Network and D-S Evidence Theory. *2012 International Conference on Industrial Control and Electronics Engineering*. doi:10.1109/icicee.2012.390

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*(2), 241-259. doi:10.1016/s0893-6080(05)80023-1

Xu, W., & Yu, J. (2017). A novel approach to information fusion in multi-source datasets: A granular computing viewpoint. *Information Sciences, 378*, 410-423. doi:10.1016/j.ins.2016.04.009

Yager, R. (1987). On the Dempster-Shafer framework and new combination rules. *Information Sciences, 41*(2), 93-137. doi:10.1016/0020-0255(87)90007-7

Yager, R. (1996). On the aggregation of prioritized belief structures. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 26*(6), 708-717. doi:10.1109/3468.541331

Ye, F., Chen, J., & Tian, Y. (2018). A Robust DS Combination Method Based on Evidence Correction and Conflict Redistribution. *Journal of Sensors, 2018*, 1-12. doi:10.1155/2018/6526018

Yong, D., Wenkang, S., Zhenfu, Z., & Qi, L. (2004). Combining belief functions based on distance of evidence. *Decision Support Systems, 38*(3), 489-493. doi:10.1016/j.dss.2004.04.015

Zadeh, L. A. (1979). On the Validity of Dempster's Rule of Combination of Evidence, ERL Memo M 79/24, University of California, Berkeley.

Zadeh, L. A. (1984). Book Review: A Mathematical Theory of Evidence. *AI Magazine, 5*(3), 81-83. doi:10.1090/S0002-9904-1977-14338-3

Zadeh, L. A. (1986). A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination. *The AI Magazine, 7*(2), 85-90.

Zanghirati, G., & Zanni, L. (2003). A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing, 29*(4), 535–551. doi: 10.1016/s0167-8191(03)00021-8

Zhang, L., Ding, L., Wu, X., & Skibniewski, M. J. (2017). An improved Dempster–Shafer approach to construction safety risk perception. *Knowledge-Based Systems, 132*, 30-46. doi:10.1016/j.knosys.2017.06.014

Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion, 42*, 146-157. doi:10.1016/j.inffus.2017.10.006

Zhang, W., Ji, X., Yang, Y., Chen, J., Gao, Z., & Qiu, X. (2018). Data Fusion Method Based on Improved D-S Evidence Theory. *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. doi:10.1109/bigcomp.2018.00145

Zhou, Q., Zhou, H., Zhou, Q., Yang, F., Luo, L., & Li, T. (2015). Structural damage detection based on posteriori probability support vector machine and Dempster–Shafer evidence theory. Applied Soft Computing, 36, 368-374. doi:10.1016/j.asoc.2015.06.057

APPENDICES

A. SCALING OPERATION OF CORRELATION COEFFICIENTS

ECMDF and ECMDFS scale the elements of correlation matrix, $CORR$, for different intervals defined in $sInterval$ in Step 2.2. In this appendix, details of the scaling operation is given.

The following notations are used for scaling.

min_i	Minimum value in row i in correlation matrix, $CORR$.
max_i	Maximum value in row i in correlation matrix, $CORR$.
$corr_{ij}$	Correlation coefficient between sensor i and j's predicted classes for all training evidences.
$sCorr_{ij}$	Scaled correlation coefficient between sensor i and j's predicted classes for all training evidences.
x_{ij}	Auxiliary variable that helps to calculate $sCorr_{ij}$.
$sInterval$	List that contains different intervals of scaling.
$u = (x, y)$	u represents the scaling interval whereas x and y shows the lower and upper bounds, respectively.

Suppose that $sInterval = \{[1,2],[1,3],[1,4],[1,5]\}$ and $CORR$ is as follows:

$$CORR = \begin{pmatrix} 1 & 0.3473 & 0.3324 \\ 0.3473 & 1 & 0.5940 \\ 0.3324 & 0.5940 & 1 \end{pmatrix} \quad \text{(A.1)}$$

According to $sInterval$, ECMDF and ECMDFS will try four different scaling intervals and decide which one gives the best solution among them based on number of correctly predicted targets.

$$min_i = \min_j \ corr_{ij} \qquad \forall i \tag{A.2}$$

$$max_i = \max_j \ corr_{ij} \qquad \forall i \tag{A.3}$$

Scaling operation is done by min-max scaler along the each row of $CORR$. Between the intervals defined in $sInterval$, we use (1,2) interval for scaling first.

$$u = (1,2) \text{ where } x = 1 \text{ and } y = 2 \tag{A.4}$$

In each row of $CORR$, min_i and max_i are determined first. According to min-max scaling, min_i is scaled as x and max_i is scaled as y. Then, the other $corr_{ij}$ values in each row are scaled according to (A.5) and (A.6).

$$x_{ij} = \frac{corr_{ij} - min_i}{max_i - min_i} \qquad \forall i,j \tag{A.5}$$

$$sCorr_{ij} = x_{ij} * (y - x) + x \qquad \forall i,j \tag{A.6}$$

To make the calculations more clear, we present the calculation of scaled correlation coefficient between the first and second sensors, $sCorr_{12}$, below in (A.7)-(A.13).

$$min_1 = 0.3324 \tag{A.7}$$

$$max_1 = 1 \tag{A.8}$$

$$x_{12} = \frac{corr_{12} - min_1}{max_1 - min_1} \tag{A.9}$$

$$sCorr_{12} = x_{12} * (y - x) + x \tag{A.10}$$

$$x_{12} = \frac{0.3473 - 0.3324}{1 - 0.3324} = 0.0223 \tag{A.11}$$

$$sCorr_{12} = 0.0223 * (2 - 1) + 1 = 1.0223 \tag{A.12}$$

After calculating all the pairwise correlation coefficients, $sCORR_{(1,2)}$ is derived.

$$sCORR_{(1,2)} = \begin{pmatrix} 2 & 1.0223 & 1 \\ 1 & 2 & 0.5939 \\ 1 & 1.3918 & 2 \end{pmatrix} \tag{A.13}$$

B. ECMDFS ALGORITHM

In this appendix, we give the details of ECMDFS. The same notations in Section 3.4 together with the following ones are used.

nm'_{kp}	Normalized unified discounted BPA of proposition p for target k for for training datasets of all sensors.
nmt'_{mp}	Normalized unified discounted BPA of proposition p for target m for for test datasets of all sensors.

The Training Phase:

Step 0. Read datasets for all sensors.

Step 1.1. Split datasets of sensors as training and test with user specified test size.

Step 1.2. Train separately both ANN and SVM for each sensor i.

Step 1.3. Probabilistically classify sensors training datasets with trained and tuned ANN and SVM.

Step 1.4. For each sensor i, if ANN gives higher classification accuracy on training dataset of sensor i, then assign its probabilistic classification results for each proposition to respective BPA of sensor i for target k for the same proposition p, m_{kip}. Assign its probabilistic classification result to sensor i's piece of evidence for target k, m_{ki}. Assign its predicted classes of each target to sensor i's predicted classes for target k, mC_{ki}. Assign its classification accuracy ratio to sensor i's accuracy ratio, acc_i. Otherwise, assign SVM's results to them.

Step 1.5. To penalize the sensors that have lower sensor accuracies and to reward the ones that have higher accuracies, take the second power of each of them and normalize them. Do it for each acc_i for each sensor i and update acc_i accordingly.

Step 2.1. By using sensor i's and sensor j's predicted classes for each target in training datasets, mC_{ki} and mC_{kj}, calculate the pairwise Pearson correlation coefficients, $corr_{ij}$. Built the correlation matrix, $CORR$.

$$corr_{ij} = \frac{\sum_{k=1}^{nbTrainingTargets}(mC_{ki} - \overline{mC}_{ki})(mC_{kj} - \overline{mC}_{kj})}{\sqrt{\sum_{k=1}^{nbTrainingTargets}(mC_{ki} - \overline{mC}_{ki})^2}\sqrt{\sum_{k=1}^{nbTrainingTargets}(mC_{kj} - \overline{mC}_{kj})^2}} \quad \forall i,j \tag{B.1}$$

$$CORR = \begin{bmatrix} corr_{11} & \cdots & corr_{1j} \\ \vdots & \ddots & \vdots \\ corr_{i1} & \cdots & corr_{ij} \end{bmatrix} \tag{B.2}$$

Where $\overline{mC}_{ki} = \frac{\sum_{l=1}^{nbTrainingTargets} mC_{li}}{nbTrainingTargets}$ and $\overline{mC}_{kj} = \frac{\sum_{l=1}^{nbTrainingTargets} mC_{lj}}{nbTrainingTargets}$.

Step 2.2. Scale each $corr_{ij}$ in correlation matrix according to the first interval in $sInterval$ and assign its result to each $sCorr_{ij}$ for sensor i and sensor j. Details of the scaling operation are given in Appendix A.

Step 2.3. For each target k, sensor i and sensor j; by using the scaled correlation coefficients in Step 2.2 as p parameter of L_p metric, calculate the distances between sensors i and j's evidences, m_{ki} and m_{kj}, for each target k by using sensors accuracies, acc_i and acc_j, as weight.

$$d_{kij} = \left(\sum_{p=1}^{nbPropositions} (acc_i * acc_j)^{sCorr_{ij}} * |m_{kip} - m_{kjp}|^{sCorr_{ij}} \right)^{\frac{1}{sCorr_{ij}}} \quad \forall k,i,j \tag{B.3}$$

Step 2.4. Built distance matrix,DIS_k for each target k.

$$DIS_k = \begin{bmatrix} 1 & \cdots & d_{k1j} \\ \vdots & \ddots & \vdots \\ d_{ki1} & \cdots & 1 \end{bmatrix} \qquad \forall k \tag{B.4}$$

Step 2.5. Normalize each DIS_k for each target k.

Step 2.6. Calculate the similarity degrees between evidences of sensor i and j, s_{kij}, for each sensor and target k and built the similarity matrix for each target k.

$$s_{kij} = 1 - d_{kij} \qquad \forall k, i, j \tag{B.5}$$

$$SIM_k = \begin{bmatrix} 1 & \cdots & s_{k1j} \\ \vdots & \ddots & \vdots \\ s_{ki1} & \cdots & 1 \end{bmatrix} \qquad \forall k \tag{B.6}$$

Step 2.7. Calculate the credibility degrees, crd_{ki} of each sensor i for each target k. Then, take the averages of credibility degrees to find average credibility degree of sensor i for all targets.

$$sup_{ki} = \sum_{j=1}^{nbSensors} s_{kij} \qquad \forall k, i \tag{B.7}$$

$$crd_{ki} = \frac{sup_{ki}}{\sum_{j=1}^{nbSensors} sup_{kj}} \qquad \forall k, i \tag{B.8}$$

$$crd_i = \frac{\sum_{k=1}^{nbTrainingTargets} crd_{ki}}{nbTrainingTargets} \qquad \forall i \tag{B.9}$$

Step 2.8. To penalize the sensors that have lower credibility degrees, and reward the ones that have higher credibility degrees, take the second power of them and normalize. Do it for each crd_i for each sensor i and update them accordingly.

Step 2.9. Discount each BPA of sensor i, target k and proposition p, m_{kip}, and sum over all the sensors to get one unified BPA , m'_{kp}, for each target k and proposition p.

$$m'_{kp} = \sum_{i=1}^{nbSensors} crd_i * m_{kip} * acc_i \qquad \forall k, p \tag{B.10}$$

Step 2.10. Normalize m'_{kp} for each target k and proposition p**.**

$$nm'_{kp} = \frac{m'_{kp}}{\sum_{h=1}^{nbPropositions} m'_{kh}} \qquad \forall k, p \tag{B.11}$$

Step 2.11. Calculate the combined conflicting degree, k_{kt}, for each target k for combination step $t = 1$.

$$k_{kt} = \sum_{H_p \cap H_r = \emptyset} nm'_{kp} * nm'_{kr} \qquad \forall k, p = 1,2, \dots, nbSensors - 1 \tag{B.12}$$

Step 2.12. Using equation (B.13), combine the unified BPA's once by using conflicting degree in (B.12). Then, go to Step 2.11, increase t by 1 and calculate the conflicting degree between the same unified evidence by combining it t many times with itself. Through the conflicting degree, combine the unified BPAs t many times. Repeat this steps $nbSensors - 1$ times for each target k and proposition p.

$$m''_{kpt} = \begin{cases} \dfrac{1}{1 - k_{kt}} \displaystyle\sum_{H_p \cap H_r = H_p} nm'_{kp} * nm'_{kr} \\ 0 \end{cases} \tag{B.13}$$

$$if\ p \neq \emptyset\ , \forall k, p, t = 1, \dots, nbSensors - 1$$
$$if\ p = \emptyset\ , \forall k, p, t = 1, \dots, nbSensors - 1$$

When $t = nbSensors - 1$, assign the value of m''_{kpt} to m''_{kp}.

Step 2.13. Calculate belief degree for each target k, and proposition p.

$$Bel(H_p)_k = \sum_{H_l \subset H_p} mt''_{kl} \qquad \forall k, p \tag{B.14}$$

Step 2.14. Turn back to Step 2.2. Scale correlation coefficient matrix, $CORR$ with the next interval in $sInterval$. Repeat the steps 2.2 to 2.13 until there is no new interval in $sInterval$.

Step 2.15. Choose the best scaling interval, u, that gives the highest number of correct predictions for test phase of ECMDF. As learnt credibility degrees, assign the credibility degrees calculated during best scaling interval to $fCrd_i$ to be used in test phase later.

The Test Phase:

Step 3.1. For each sensor, with the selected classifier in Step 1.4, probabilistically classify sensor i's test dataset. Assign its probabilistic classification results for each proposition to respective BPA of sensor i for target m for the same proposition p, mt_{mip}.

Step 3.2. With normalized **c**redibility degrees determined in training phase in Step 2.15, $fCrd_i$, and sensor accuracies, acc_i, in Step 2.8, discount each BPA, mt_{mip}, for each sensor i, proposition p and target m and sum over all the sensors to get one unified BPA , mt'_{mp}, for each proposition p and target m.

$$mt'_{mp} = \sum_{i=1}^{nbSensors} fCrd_i * mt_{mip} * acc_i \qquad \forall m, p \tag{B.15}$$

Step 3.3. Normalize mt'_{mp} for each target m and proposition p**.**

$$nmt'_{mp} = \frac{mt'_{mp}}{\sum_{m=1}^{nbPropositions} mt'_{mp}} \qquad \forall m, p \tag{B.16}$$

Step 3.4. Calculate combined conflicting degree, k_{mt}, for each target m for combination step $t = 1$.

$$k_{mt} = \sum_{H_p \cap H_r = \emptyset} nmt'_{mp} * nmt'_{mr} \qquad \forall m, t = 1,2, \dots, nbSensors - 1 \tag{B.17}$$

Step 3.5. Using equation (B.18), combine the unified BPA's once by using conflicting degree in (B.17). Then, go to Step 3.4, increase t by 1 and calculate the conflicting degree between the same unified evidences by combining it t many times with itself. Through the conflicting degree, combine the unified BPAs t many times. Repeat this steps $nbSensors - 1$ times for each target m and proposition p.

$$mt''_{mpt} = \begin{cases} \dfrac{1}{1 - k_{mt}} \displaystyle\sum_{H_p \cap H_r = H_p} nmt'_{mp} * nmt'_{mr} \\ 0 \end{cases} \tag{B.18}$$

$$if\ p \neq \emptyset, \forall m, p, t = 1, \dots, nbSensors - 1$$
$$if\ p = \emptyset, \forall m, p, t = 1, \dots, nbSensors - 1$$

when $t = nbSensors - 1$, assign the value of mt''_{mpt} to mt''_{mp}.

Step 3.6. Calculate belief degree for each target m, and proposition p.

$$Bel(H_p)_m = \sum_{H_l \subset H_p} mt''_{ml} \qquad \forall m, p \tag{B.19}$$

Step 3.7. Assign the class of proposition p that has the highest belief degree, $Bel(H_p)_m$, to target m as the result of the fusion process.

C. DEFAULT HYPER PARAMETERS OF CLASSIFIERS

Table C.1. *Default parameters used in training of each ANN for each sensor*

ANN Parameters		**Default Values**
Weights Initialization	Kernel Initializer	Random Normal
	Bias Initializer	Random Normal
Number of Hidden Layers		1
Number of Hidden Neurons[1]		$(f + \sqrt{s})/L$
Activation Function		Rectified Linear Unit
Learning Rate		0.01
Optimizer		Stochastic Gradient Descent
Loss Function		Categorical Cross Entropy
Number of Epochs		32
Batch Size		50
Number of Iterations for Random Search		5

[1] f represents number of features, s represents number of samples and L is the number of hidden layer.

Table C.2. *Default parameters used in training of each SVM for each sensor*

SVM Parameters	**Default Values**
Kernel Function	Radial Basis Function
C	1
Gamma	$\frac{1}{Number\ of\ Features}$
Decision Function Shape	One versus rest

D. HYPER PARAMETERS OF ARTIFICIAL NEURAL NETWORK

In this appendix, hyper parameters of ANN are presented in more detail.

Activation Function

Activation function determines the output of each node by the help of the bias added input multiplied by weights. According to Géron (2018), there are three popular activation functions in terms of performance, which are sigmoid, hyperbolic tangent and rectified linear unit activation function. We adopt these three activation functions to tune the ANN.

Sigmoid Activation Function: Sigmoid Activation Function maps the input between 0 and 1 according to the formula (D.1). By this property, it avoids unreasonably large output values. However, as the outputs are not centered on zero, it leads gradients to be always either positive or negative during backpropagation, which cause slower convergence.

$$sig(z) = \frac{1}{(1 + e^{z})} \tag{D.1}$$

In Figure D.1, output of sigmoid activation function for different z values is presented.

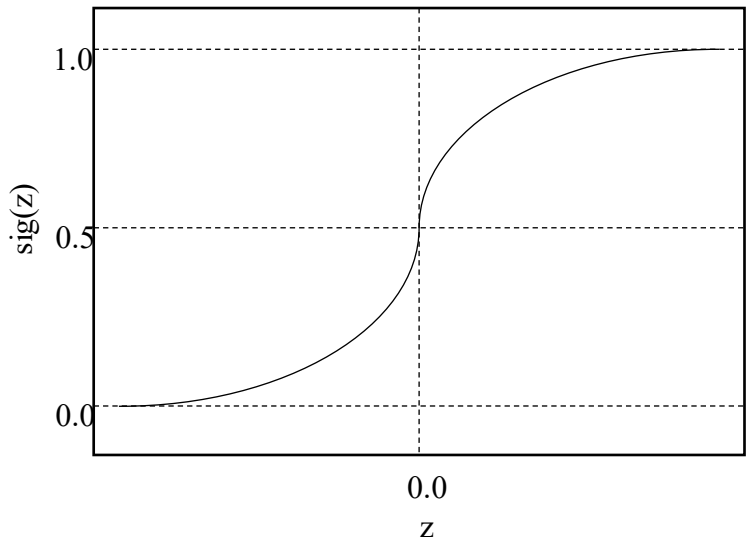

Figure D.1. Output of sigmoid activation function for different z values

Hyperbolic Tangent Activation Function: Hyperbolic tangent activation function maps the input between -1 and 1 according to (D.2). As opposed to sigmoid activation function, outputs are centered on 0.5, which leads faster convergence during backpropagation.

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{D.2}$$

In Figure D.2, output of hyperbolic activation function for different z values is presented.

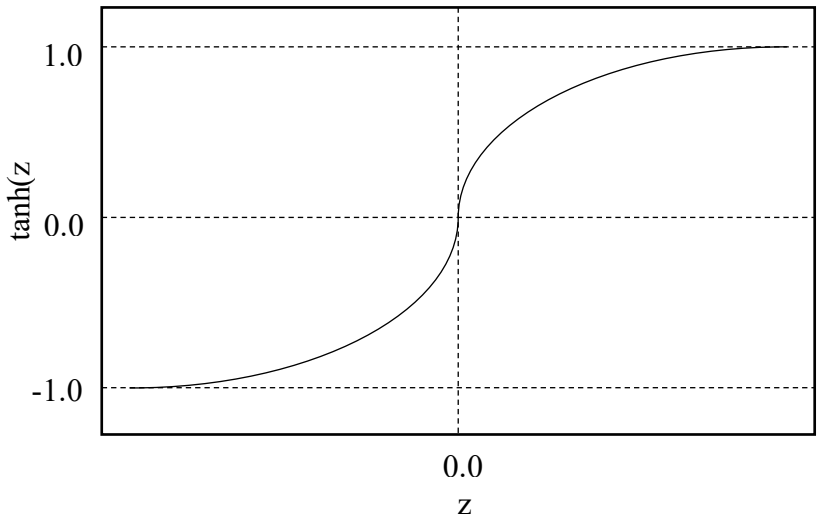

Figure D.2. Output of hyperbolic activation function for different z values

Rectified Linear Unit Activation Function: This activation function is popular in ANN as it has greater convergence speed than sigmoid and hyperbolic tangent activation function. Its formula can be seen from (D.3).

It gives the output of the corresponding node as either the input itself if the input value is positive, and zero if the input is negative according to (D.3). By doing so, it avoids outputs to increase exponentially. In addition, it is fast.

$$Relu(z) = \max(0, z) \tag{D.3}$$

In Figure D.3, output of rectified linear activation function for different z values is presented.

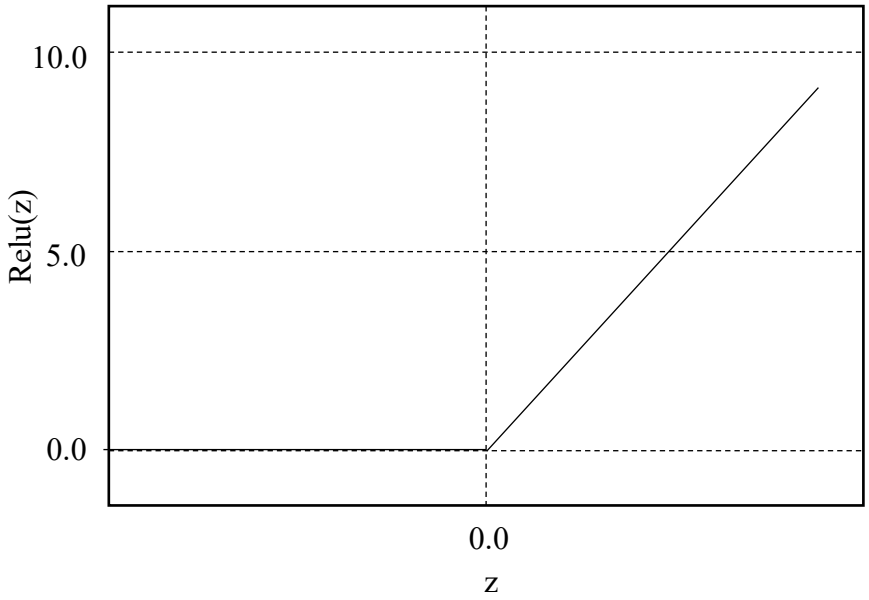

Figure D.3. Output of rectified linear activation function for different *z* values

Optimizer

In ANN, loss functions help us to calculate the difference between predicted value and actual value in each iteration. On the other hand, optimizers help us to calculate the derivative, *gradient*, and decrease the error in each iteration and hence, minimize the overall error. Stochastic gradient descent, root mean square propagation, adaptive gradient and adaptive moment estimation optimizers are among the most popular optimization algorithms in ANN (Géron, 2018). We use these optimizers in random search for tuning.

E. HYPER PARAMETERS OF SUPPORT VECTOR MACHINE

In this appendix, hyper parameters of SVM are presented in more detail.

Cost Parameter, C

Cost parameter, C, is the user specified cost that allows soft margin to classify some of the data points as wrong classes. It ensures more flexible model. The effects of *C* parameter can be seen from Figure E.1.

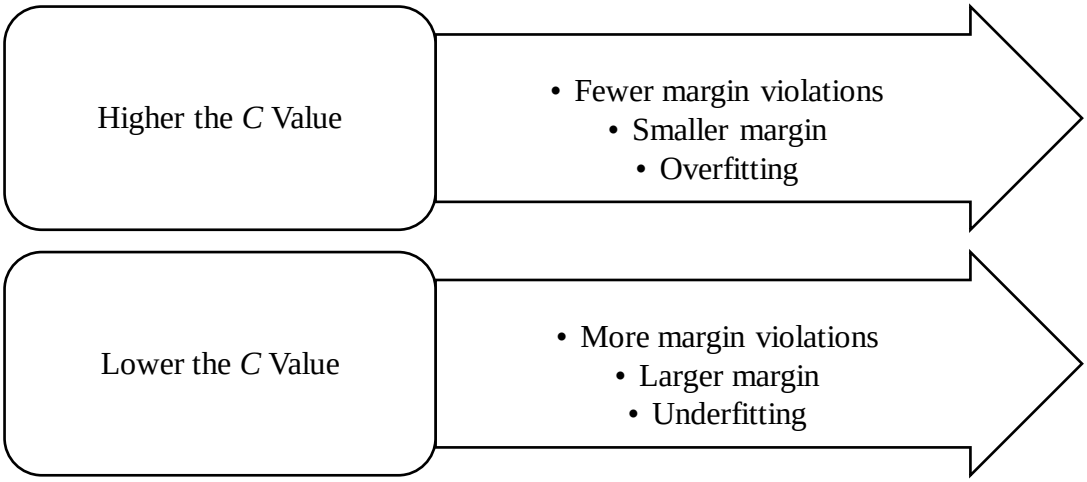

Figure E.1. Effect of different values of C

In the computational study, $[1,5,10]$ is used to try different levels of *C*.

Kernel function

Géron (2018) states that polynomial and radial basis function are the most popular kernel functions in SVM. Thus, both of them are used in grid search in SVM.

Polynomial Kernel Function: Polynomial kernel has three parameters to tune, which are d, γ and r. As the number of parameters is high, this type of kernel may take longer time than radial basis function.

$$K(x, y) = (\gamma * (x, y) + r)^d \quad \text{(E.1)}$$

where d is the degree, γ is the gamma and r is the independent term.

d determines the degree of the function. As d and γ increase, classification performance of the SVM gets higher. However, we may encounter the overfitting problem. As they decreases, generalization performance of the SVM increases. Yet, we may encounter underfitting problem. r is the free parameter of the polynomial kernel function. While tuning the parameters of the kernel, the value of γ is determined automatically by the SVM and r is taken as zero.

The following different values of d are used in tuning.

$d = [1,2,3]$

Radial Basis Function: Radial basis function has two parameters, which are C and γ as can be seen from (E.2).

$$K(x, y) = e^{-\gamma * ||x-y||^2 + C} \quad \text{(E.2)}$$

Radial basis function takes less time than polynomial kernel function. C is the cost

parameter, and γ is the regularization parameter. As the value of γ increases, decision boundary gets more irregular. On the other hand, as the value of γ decreases, decision boundary gets smoother. Large values of this parameter may lead overfitting problem, and small values of it may lead under fitting problem. In the computational experiments, the value of it is determined automatically in tuning phase.

F. DETAILED EXPERIMENTAL RESULTS OF MULTI-SENSOR TARGET CLASSIFICATION MODELS

Table F.1. *Classification accuracy of ANN and SVM for datasets generated with first seed*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	87	87	87	87	91	91
5/500/0	84	89	90	84	92	91
5/1000/0	83	88	88	84	94	93
5/100/1	87	87	34	87	91	34
5/500/1	84	89	41	84	92	36
5/1000/1	83	88	37	84	94	32
5/100/2	87	36	34	87	36	34
5/500/2	84	38	41	84	35	36
5/1000/2	83	35	37	84	33	32
10/100/0	100	97	82	94	100	79
10/500/0	89	94	72	96	99	69
10/1000/0	93	94	79	97	97	83
10/100/1	100	97	34	94	100	27
10/500/1	89	94	45	96	99	33
10/1000/1	93	94	33	97	97	35
10/100/2	100	57	34	94	27	27
10/500/2	89	48	45	96	34	33
10/1000/2	93	35	33	97	35	35
Overall	89	75	53	90	75	50

Table F.*2. Computational results for datasets generated with first seed*

Problem Setting	ECDST				ECY				ECMDF				ECMDFS			
	PCP	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**
5/100/0	87.88	0.1394	0.1745	-0.1151	90.91	0.1403	0.1729	-0.1862	90.91	0.1416	0.1717	-0.1597	90.91	0.1401	0.1698	-0.1570
5/500/0	90.91	0.0857	0.1134	-0.1913	92.12	0.0953	0.1226	-0.2240	90.91	0.0915	0.1196	-0.1892	91.52	0.0918	0.1199	-0.2112
5/1000/0	88.79	0.0917	0.1168	-0.1068	88.48	0.0985	0.1248	-0.1038	89.09	0.0971	0.1208	-0.0966	88.79	0.0979	0.1220	-0.0930
5/100/1	90.91	0.2563	0.2846	-0.0264	87.88	0.2657	0.3097	-0.0535	90.91	0.2294	0.2575	-0.0517	90.91	0.2653	0.3001	-0.0830
5/500/1	91.52	0.2427	0.2758	-0.1137	90.30	0.2285	0.2662	-0.1227	91.52	0.2298	0.2639	-0.1375	90.91	0.2461	0.2854	-0.1463
5/1000/1	86.97	0.2290	0.2843	-0.1399	84.85	0.2121	0.2734	-0.1313	85.76	0.2141	0.2730	-0.1405	85.76	0.2275	0.2887	-0.1404
5/100/2	87.88	0.2849	0.3229	0.0091	33.33	-0.1358	0.0851	-0.2463	30.30	-0.0582	0.1695	-0.1571	81.82	0.2265	0.2791	-0.0106
5/500/2	82.42	0.2575	0.3164	-0.0187	33.94	-0.1333	0.0075	-0.2057	55.15	-0.0396	0.0147	-0.1063	84.24	0.1896	0.2345	-0.0504
5/1000/2	79.39	0.2489	0.3438	-0.1167	34.24	-0.1097	-0.0301	-0.1511	79.09	-0.0203	-0.0212	-0.0173	79.39	0.2142	0.3001	-0.1168
10/100/0	72.73	0.0816	0.1691	-0.1518	72.73	0.0900	0.1665	-0.1140	72.73	0.0873	0.1700	-0.1330	72.73	0.0917	0.1736	-0.1268
10/500/0	90.30	0.1578	0.1972	-0.2096	88.48	0.1452	0.1928	-0.2211	89.70	0.1652	0.1966	-0.1078	92.12	0.1746	0.2008	-0.1320
10/1000/0	88.79	0.1296	0.1655	-0.1547	87.27	0.1177	0.1629	-0.1925	88.79	0.1293	0.1618	-0.1280	88.79	0.1301	0.1625	-0.1271
10/100/1	75.76	0.1856	0.3056	-0.1896	78.79	0.1676	0.2315	-0.0697	75.76	0.1831	0.3017	-0.1877	75.76	0.1881	0.3123	-0.2000
10/500/1	93.33	0.2426	0.2724	-0.1743	90.30	0.2319	0.2685	-0.1087	92.12	0.2482	0.2805	-0.1293	93.33	0.2560	0.2863	-0.1681
10/1000/1	89.70	0.2283	0.2717	-0.1498	88.18	0.1989	0.2400	-0.1079	89.09	0.2245	0.2681	-0.1310	89.09	0.2292	0.2745	-0.1413
10/100/2	75.76	0.2526	0.3743	-0.1274	60.61	0.0191	0.0631	-0.0487	75.76	0.0455	0.0479	0.0381	75.76	0.2123	0.3160	-0.1119
10/500/2	81.21	0.2446	0.3224	-0.0917	37.58	-0.0796	0.1310	-0.2064	44.24	-0.0440	0.0924	-0.1523	80.00	0.1759	0.2395	-0.0785
10/1000/2	85.45	0.2968	0.3628	-0.0908	38.18	-0.1357	-0.0968	-0.1596	80.61	-0.0292	-0.0299	-0.0264	86.06	0.2827	0.3504	-0.1349
Overall	85.54	0.2031	0.2596	-0.1200	71.01	0.0787	0.1495	-0.1474	78.47	0.1053	0.1588	-0.1118	85.44	0.1911	0.2453	-0.1239

Table F.3. *Chosen scaling intervals for datasets generated with first seed*

Problem Setting	**ECMDF**	**ECMDFS**
5/100/0	[1, 2]	[1, 2]
5/500/0	[1, 2]	[1, 2]
5/1000/0	[1, 2]	[1, 2]
5/100/1	[1, 2]	[1, 2]
5/500/1	[1, 20]	[1, 40]
5/1000/1	[1, 40]	[1, 2]
5/100/2	[1, 2]	[1, 2]
5/500/2	[1, 40]	[1, 2]
5/1000/2	[1, 2]	[1, 2]
10/100/0	[1, 2]	[1, 2]
10/500/0	[1, 2]	[1, 2]
10/1000/0	[1, 2]	[1, 2]
10/100/1	[1, 2]	[1, 2]
10/500/1	[1, 2]	[1, 5]
10/1000/1	[1, 2]	[1, 2]
10/100/2	[1, 2]	[1, 2]
10/500/2	[1, 40]	[1, 40]
10/1000/2	[1, 2)	[1, 2]

Table F.4. *Computational times for datasets generated with first seed*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	97	4	144	6	152	5	151	4
5/500/0	208	9	199	12	242	7	246	6
5/1000/0	230	13	217	17	294	12	305	9
5/100/1	186	7	176	8	191	7	198	5
5/500/1	331	10	321	12	364	8	372	6
5/1000/1	766	12	752	19	840	12	848	11
5/100/2	198	6	190	7	201	6	208	5
5/500/2	494	10	481	14	519	7	535	7
5/1000/2	1330	13	1322	16	1394	14	1424	11
10/100/0	199	8	189	10	203	8	211	6
10/500/0	219	10	211	12	267	10	279	10
10/1000/0	311	17	295	20	386	14	400	12
10/100/1	201	8	188	9	203	9	214	5
10/500/1	425	11	411	14	465	9	473	7
10/1000/1	643	13	629	19	719	12	726	10
10/100/2	192	9	178	8	194	7	202	4
10/500/2	447	12	436	12	480	9	494	9
10/1000/2	1699	16	1690	17	1769	13	1786	13
Overall	454	10	446	13	494	9	504	8

Table F.5. *Classification accuracy of ANN and SVM for datasets generated with second seed*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	78	64	60	93	84	87
5/500/0	87	80	70	94	87	83
5/1000/0	83	77	89	87	80	93
5/100/1	78	64	42	93	84	36
5/500/1	87	80	35	94	87	35
5/1000/1	83	77	36	87	80	32
5/100/2	78	40	42	93	40	36
5/500/2	87	31	35	94	34	35
5/1000/2	83	37	36	87	35	32
10/100/0	70	54	81	100	88	100
10/500/0	93	63	92	97	63	98
10/1000/0	92	74	94	97	71	94
10/100/1	70	54	34	100	88	31
10/500/1	93	63	40	97	63	35
10/1000/1	92	74	44	97	71	30
10/100/2	70	49	34	100	28	31
10/500/2	93	33	40	97	34	35
10/1000/2	92	45	44	97	29	30
Overall	83.83	58.83	52.67	94.67	63.67	52.94

Table F.6. *Computational results for datasets generated with second seed*

Problem Setting	ECDST				ECY				ECMDF				ECMDFS			
	PCP	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**
5/100/0	87.88	0.2165	0.2672	-0.1512	87.88	0.2021	0.2612	-0.2263	90.91	0.2052	0.2445	-0.1871	90.91	0.2127	0.2523	-0.1830
5/500/0	95.15	0.1492	0.1646	-0.1515	93.94	0.1458	0.1673	-0.1869	94.55	0.1354	0.1503	-0.1234	93.33	0.1327	0.1522	-0.1395
5/1000/0	91.52	0.1588	0.1864	-0.1387	90.91	0.1491	0.1818	-0.1782	92.12	0.1466	0.1701	-0.1280	92.42	0.1496	0.1729	-0.1354
5/100/1	81.82	0.2582	0.3321	-0.0744	81.82	0.2216	0.3047	-0.1520	81.82	0.1326	0.1837	-0.0975	84.85	0.2423	0.3110	-0.1422
5/500/1	92.73	0.2620	0.2886	-0.0776	90.91	0.2348	0.2666	-0.0840	92.73	0.2449	0.2694	-0.0669	92.12	0.2575	0.2861	-0.0770
5/1000/1	89.39	0.2610	0.3064	-0.1212	77.27	0.1884	0.2770	-0.1130	87.58	0.1910	0.2285	-0.0729	89.09	0.2521	0.2970	-0.1145
5/100/2	69.70	0.1717	0.2708	-0.0561	45.45	-0.0660	0.2300	-0.3127	33.33	-0.0584	0.1324	-0.1538	78.79	0.1886	0.2344	0.0187
5/500/2	91.52	0.3222	0.3626	-0.1136	33.33	-0.1588	-0.1181	-0.1792	89.09	-0.0328	-0.0343	-0.0207	91.52	0.3128	0.3561	-0.1541
5/1000/2	83.64	0.2638	0.3235	-0.0413	38.79	-0.1292	-0.0953	-0.1507	80.00	-0.0278	-0.0282	-0.0260	83.33	0.2323	0.2915	-0.0641
10/100/0	78.79	0.1754	0.2315	-0.0330	72.73	0.0719	0.1279	-0.0773	78.79	0.1371	0.1954	-0.0791	78.79	0.1488	0.2116	-0.0841
10/500/0	84.85	0.1733	0.2234	-0.1072	81.82	0.1547	0.2183	-0.1313	84.24	0.1666	0.2210	-0.1245	83.64	0.1659	0.2248	-0.1347
10/1000/0	90.30	0.1405	0.1709	-0.1428	87.58	0.1256	0.1683	-0.1753	89.09	0.1320	0.1620	-0.1127	89.09	0.1328	0.1640	-0.1219
10/100/1	66.67	0.0690	0.1143	-0.0217	30.30	-0.0348	-0.0225	-0.0402	57.58	-0.0093	-0.0082	-0.0107	66.67	0.0319	0.0556	-0.0155
10/500/1	83.03	0.2654	0.3403	-0.1007	69.70	0.0696	0.1687	-0.1584	70.30	0.0667	0.1364	-0.0981	80.61	0.2177	0.2891	-0.0789
10/1000/1	84.55	0.2423	0.3090	-0.1228	76.97	0.1571	0.2515	-0.1584	78.79	0.1529	0.2283	-0.1270	84.55	0.2206	0.2839	-0.1259
10/100/2	54.55	0.0609	0.1374	-0.0308	42.42	0.0029	0.0726	-0.0485	33.33	-0.0071	0.0655	-0.0434	57.58	0.0734	0.1502	-0.0309
10/500/2	83.03	0.2762	0.3484	-0.0773	32.12	-0.1293	-0.0357	-0.1736	64.85	-0.0421	-0.0132	-0.0952	81.82	0.2527	0.3291	-0.0911
10/1000/2	86.36	0.3078	0.3738	-0.1102	43.03	-0.1077	0.0395	-0.2188	58.79	-0.0355	0.0460	-0.1517	86.36	0.2223	0.2726	-0.0965
Overall	83.08	0.2097	0.2639	-0.0929	65.39	0.0610	0.1369	-0.1536	75.44	0.0832	0.1305	-0.0955	83.64	0.1915	0.2408	-0.0984

Table F.7. *Chosen scaling intervals for datasets generated with second seed*

Problem Setting	**ECMDF**	**ECMDFS**
5/100/0	[1, 2]	[1, 2]
5/500/0	[1, 5]	[1, 20]
5/1000/0	[1, 40]	[1, 5]
5/100/1	[1, 2]	[1, 2]
5/500/1	[1, 20]	[1, 10]
5/1000/1	[1, 2]	[1, 2]
5/100/2	[1, 10]	[1, 40]
5/500/2	[1, 2]	[1, 2]
5/1000/2	[1, 2]	[1, 2]
10/100/0	[1, 2]	[1, 2]
10/500/0	[1, 2]	[1, 2]
10/1000/0	[1, 2]	[1, 2]
10/100/1	[1, 5]	[1, 2]
10/500/1	[1, 20]	[1, 40]
10/1000/1	[1, 40]	[1, 40]
10/100/2	[1, 2]	[1, 2]
10/500/2	[1, 2]	[1, 2]
10/1000/2	[1, 10]	[1, 2]

Table F.8. *Computational times for datasets generated with second seed*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	115	4	130	6	136	5	167	4
5/500/0	194	8	185	12	227	7	234	7
5/1000/0	214	14	204	17	279	11	288	9
5/100/1	184	8	177	8	189	7	195	7
5/500/1	359	10	346	12	393	8	399	7
5/1000/1	568	17	556	18	637	13	653	13
5/100/2	180	5	175	9	188	6	190	4
5/500/2	612	9	600	11	652	8	650	8
5/1000/2	725	13	714	17	768	12	804	11
10/100/0	190	9	176	9	192	7	202	6
10/500/0	216	10	205	14	256	9	260	7
10/1000/0	289	17	274	20	364	13	377	13
10/100/1	195	9	180	9	195	9	206	5
10/500/1	420	12	403	15	459	10	468	7
10/1000/1	2268	14	2268	14	2337	12	2345	12
10/100/2	199	6	188	11	205	7	213	5
10/500/2	833	12	822	11	868	9	883	9
10/1000/2	2827	12	2817	18	2898	13	2907	10
Overall	588	11	579	13	625	9	636	8

Table F.9. *Classification accuracy of ANN and SVM for datasets generated with third seed*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	73	78	75	100	91	81
5/500/0	88	77	58	99	85	79
5/1000/0	83	66	65	91	74	69
5/100/1	73	78	40	100	91	40
5/500/1	88	77	36	99	85	35
5/1000/1	83	66	36	91	74	36
5/100/2	73	39	40	100	31	40
5/500/2	88	35	36	99	34	35
5/1000/2	83	33	36	91	32	36
10/100/0	81	84	81	100	100	100
10/500/0	70	69	72	98	96	99
10/1000/0	71	71	72	95	94	100
10/100/1	81	84	61	100	100	33
10/500/1	70	69	35	98	96	33
10/1000/1	71	71	37	95	94	33
10/100/2	81	45	61	100	31	33
10/500/2	70	39	35	98	34	33
10/1000/2	71	37	37	95	33	33
Overall	78	62	51	97	71	53

Table F.30. *Computational results for datasets generated with third seed*

Problem Setting	ECDST				ECY				ECMDF				ECMDFS			
	PCP	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**
5/100/0	72.73	0.1572	0.2492	-0.0881	78.79	0.1533	0.2232	-0.1062	72.73	0.1505	0.2449	-0.1013	72.73	0.1540	0.2521	-0.1076
5/500/0	94.55	0.1693	0.1829	-0.0667	93.94	0.1458	0.1683	-0.2023	93.33	0.1232	0.1408	-0.1244	93.94	0.1389	0.1510	-0.0479
5/1000/0	86.97	0.1844	0.2396	-0.1839	81.52	0.1311	0.2068	-0.2029	86.36	0.1506	0.1980	-0.1493	85.45	0.1668	0.2218	-0.1563
5/100/1	72.73	0.1985	0.3122	-0.1047	69.70	0.1810	0.2845	-0.0571	72.73	0.1578	0.2589	-0.1118	72.73	0.2112	0.3443	-0.1439
5/500/1	94.55	0.2703	0.2923	-0.1118	85.45	0.1904	0.2560	-0.1947	88.48	0.2050	0.2482	-0.1267	93.94	0.2496	0.2716	-0.0924
5/1000/1	85.45	0.2503	0.3114	-0.1089	79.70	0.1821	0.2562	-0.1090	83.03	0.1702	0.2207	-0.0770	84.85	0.2355	0.2975	-0.1116
5/100/2	78.79	0.2080	0.2718	-0.0290	48.48	-0.0853	-0.0301	-0.1372	78.79	-0.0245	-0.0200	-0.0409	81.82	0.2072	0.2625	-0.0414
5/500/2	96.36	0.3926	0.4126	-0.1387	30.91	-0.1945	0.0015	-0.2822	69.09	-0.0578	-0.0096	-0.1654	96.36	0.3557	0.3746	-0.1439
5/1000/2	83.64	0.2974	0.3679	-0.0633	35.76	-0.1209	-0.0288	-0.1721	75.45	-0.0293	-0.0189	-0.0613	84.24	0.2760	0.3501	-0.1205
10/100/0	78.79	0.1371	0.2024	-0.1053	84.85	0.1601	0.2070	-0.1030	81.82	0.1389	0.1876	-0.0803	81.82	0.1389	0.1876	-0.0803
10/500/0	89.70	0.1189	0.1471	-0.1262	90.91	0.1237	0.1503	-0.1426	89.70	0.1124	0.1368	-0.0996	90.30	0.1115	0.1356	-0.1127
10/1000/0	89.70	0.1178	0.1458	-0.1255	87.27	0.1140	0.1530	-0.1539	89.39	0.1057	0.1283	-0.0850	89.39	0.1063	0.1290	-0.0848
10/100/1	75.76	0.1964	0.2883	-0.0905	75.76	0.1967	0.2820	-0.0698	75.76	0.2070	0.3128	-0.1235	75.76	0.2119	0.3221	-0.1326
10/500/1	88.48	0.2386	0.2860	-0.1264	85.45	0.2073	0.2635	-0.1228	88.48	0.2373	0.2864	-0.1396	88.48	0.2441	0.2952	-0.1488
10/1000/1	86.97	0.2270	0.2795	-0.1233	81.21	0.1884	0.2543	-0.0963	86.67	0.2271	0.2788	-0.1089	86.36	0.2362	0.2911	-0.1113
10/100/2	75.76	0.2055	0.2971	-0.0807	36.36	-0.0561	0.1326	-0.1639	42.42	-0.0568	0.1077	-0.1781	54.55	0.0782	0.2076	-0.0771
10/500/2	83.03	0.2666	0.3355	-0.0709	52.12	-0.0629	0.0368	-0.1714	66.06	-0.0144	0.0289	-0.0988	83.64	0.2541	0.3260	-0.1137
10/1000/2	83.33	0.2625	0.3313	-0.0817	34.24	-0.1344	-0.0339	-0.1868	55.45	-0.0342	0.0114	-0.0911	83.33	0.2443	0.3117	-0.0927
Overall	84.29	0.2166	0.2752	-0.1014	68.47	0.0733	0.1546	-0.1486	77.54	0.0983	0.1523	-0.1090	83.32	0.2011	0.2629	-0.1066

Table F.11. *Chosen scaling intervals for datasets generated with third seed*

Problem Setting	**ECMDF**	**ECMDFS**
5/100/0	[1, 2]	[1, 2]
5/500/0	[1, 20]	[1, 20]
5/1000/0	[1, 20]	[1, 20]
5/100/1	[1, 2]	[1, 2]
5/500/1	[1, 2]	[1, 40]
5/1000/1	[1, 5]	[1, 40]
5/100/2	[1, 2]	[1, 2]
5/500/2	[1, 2]	[1, 2]
5/1000/2	[1, 2]	[1, 2]
10/100/0	[1, 2]	[1, 2]
10/500/0	[1, 5]	[1, 2]
10/1000/0	[1, 10]	[1, 5]
10/100/1	[1, 2]	[1, 2]
10/500/1	[1, 2]	[1, 2]
10/1000/1	[1, 20]	[1, 20]
10/100/2	[1, 2]	[1, 2]
10/500/2	[1, 5]	[1, 2]
10/1000/2	[1, 2]	[1, 2]

Table F.12. *Computational times for datasets generated with third seed*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	128	4	138	6	162	10	145	5
5/500/0	186	9	177	11	218	8	227	6
5/1000/0	217	13	209	17	283	11	292	9
5/100/1	177	9	165	8	179	6	187	5
5/500/1	741	11	729	12	781	10	791	7
5/1000/1	2043	15	2012	18	2102	13	2116	12
5/100/2	178	7	170	7	182	7	190	4
5/500/2	922	8	911	10	972	6	956	7
5/1000/2	2881	9	2860	15	2610	14	2953	9
10/100/0	186	8	168	8	186	7	194	5
10/500/0	203	12	191	13	238	9	249	7
10/1000/0	247	16	228	20	323	14	337	12
10/100/1	190	8	175	8	190	7	199	4
10/500/1	247	12	232	15	286	9	296	7
10/1000/1	417	12	401	18	500	13	506	11
10/100/2	172	6	166	8	178	7	185	4
10/500/2	330	12	317	12	362	9	376	9
10/1000/2	3510	15	3526	15	3590	13	3599	14
Overall	721	10	710	12	741	10	767	8

Table F.13. *Classification accuracy of ANN and SVM for datasets generated with fourth seed*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	76	67	73	87	69	78
5/500/0	84	63	89	84	57	91
5/1000/0	91	70	94	91	72	95
5/100/1	76	67	30	87	69	40
5/500/1	84	63	34	84	57	30
5/1000/1	91	70	35	91	72	31
5/100/2	76	33	30	87	31	40
5/500/2	84	37	34	84	35	30
5/1000/2	91	35	35	91	34	31
10/100/0	93	97	87	87	91	96
10/500/0	84	91	89	91	96	95
10/1000/0	89	87	80	88	92	85
10/100/1	93	97	52	87	91	39
10/500/1	84	91	50	91	96	30
10/1000/1	89	87	43	88	92	33
10/100/2	93	52	52	87	39	39
10/500/2	84	39	50	91	33	30
10/1000/2	89	35	43	88	33	33
Overall	86	66	56	88	64	53

Table F.4. *Computational results for datasets generated with fourth seed*

Problem Setting	ECDST				ECY				ECMDF				ECMDFS			
	PCP	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**
5/100/0	78.79	0.1643	0.2550	-0.1725	75.76	0.1541	0.2231	-0.0614	81.82	0.1723	0.2385	-0.1256	78.79	0.1663	0.2387	-0.1029
5/500/0	92.12	0.2142	0.2451	-0.1475	86.67	0.1558	0.2189	-0.2539	90.91	0.1849	0.2128	-0.0940	91.52	0.1983	0.2278	-0.1197
5/1000/0	95.15	0.1718	0.1871	-0.1295	90.00	0.1264	0.1639	-0.2114	95.45	0.1591	0.1709	-0.0887	95.76	0.1625	0.1742	-0.1023
5/100/1	75.76	0.1899	0.2847	-0.1064	75.76	0.1969	0.2894	-0.0925	78.79	0.1869	0.2507	-0.0500	75.76	0.2242	0.3244	-0.0888
5/500/1	83.03	0.2670	0.3389	-0.0845	73.33	0.0898	0.1646	-0.1160	81.21	0.1154	0.1614	-0.0831	82.42	0.2247	0.2949	-0.1047
5/1000/1	91.21	0.2987	0.3375	-0.1035	75.76	0.1350	0.2209	-0.1337	86.97	0.1525	0.1906	-0.1019	89.39	0.2594	0.3027	-0.1059
5/100/2	72.73	0.1414	0.2206	-0.0699	30.30	-0.0705	0.0080	-0.1046	39.39	-0.0247	0.0239	-0.0562	72.73	0.1449	0.2279	-0.0764
5/500/2	76.97	0.2357	0.3137	-0.0251	33.94	-0.1180	-0.0595	-0.1480	61.82	-0.0337	-0.0045	-0.0809	76.97	0.1998	0.2752	-0.0522
5/1000/2	89.39	0.3396	0.3861	-0.0524	33.94	-0.1647	-0.1263	-0.1844	87.58	-0.0316	-0.0316	-0.0316	89.70	0.3011	0.3452	-0.0835
10/100/0	87.88	0.1880	0.2369	-0.1664	87.88	0.1853	0.2361	-0.1834	81.82	0.1673	0.2247	-0.0914	84.85	0.1719	0.2231	-0.1154
10/500/0	89.70	0.1795	0.2169	-0.1461	86.67	0.1422	0.1926	-0.1855	87.88	0.1516	0.1888	-0.1180	87.88	0.1520	0.1894	-0.1187
10/1000/0	86.36	0.1401	0.1836	-0.1353	83.03	0.1124	0.1795	-0.2158	85.45	0.1260	0.1648	-0.1021	85.15	0.1270	0.1670	-0.1022
10/100/1	81.82	0.2052	0.2626	-0.0533	75.76	0.1336	0.2373	-0.1905	84.85	0.1804	0.2347	-0.1238	87.88	0.2067	0.2591	-0.1738
10/500/1	84.24	0.2540	0.3203	-0.1002	80.61	0.1702	0.2492	-0.1577	82.42	0.1829	0.2382	-0.0766	86.06	0.2358	0.2931	-0.1177
10/1000/1	85.15	0.2237	0.2894	-0.1526	80.91	0.1839	0.2584	-0.1319	83.94	0.2112	0.2719	-0.1062	83.64	0.2254	0.2913	-0.1115
10/100/2	75.76	0.2655	0.3889	-0.1202	51.52	-0.0072	0.2144	-0.2427	45.45	-0.0271	0.2002	-0.2165	69.70	0.1163	0.2141	-0.1087
10/500/2	71.52	0.1631	0.2475	-0.0486	41.21	-0.0586	0.0543	-0.1378	49.70	-0.0167	0.0608	-0.0932	72.73	0.1369	0.2115	-0.0622
10/1000/2	77.88	0.2687	0.3701	-0.0880	34.55	-0.1129	-0.0068	-0.1689	60.61	-0.0362	0.0110	-0.1088	78.48	0.2088	0.2930	-0.0984
Overall	83.08	0.2173	0.2825	-0.1057	66.53	0.0696	0.1510	-0.1622	75.89	0.1011	0.1560	-0.0972	82.74	0.1923	0.2529	-0.1025

Table F.15. *Chosen scaling intervals for datasets generated with fourth seed*

Problem Setting	**ECMDF**	**ECMDFS**
5/100/0	[1, 2]	[1, 2]
5/500/0	[1, 10]	[1, 10]
5/1000/0	[1, 5]	[1, 40]
5/100/1	[1, 5]	[1, 2]
5/500/1	[1, 10]	[1, 2]
5/1000/1	[1, 40]	[1, 40]
5/100/2	[1, 2]	[1, 2]
5/500/2	[1, 2]	[1, 2]
5/1000/2	[1, 2]	[1, 2]
10/100/0	[1, 2]	[1, 2]
10/500/0	[1, 2]	[1, 2]
10/1000/0	[1, 40]	[1, 10]
10/100/1	[1, 2]	[1, 2]
10/500/1	[1, 2]	[1, 2]
10/1000/1	[1, 2]	[1, 40]
10/100/2	[1, 2]	[1, 20]
10/500/2	[1, 40]	[1, 20]
10/1000/2	[1, 2]	[1, 2]

Table F.56. *Computational times for datasets generated with fourth seed*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	104	4	142	6	148	27	145	4
5/500/0	210	8	200	12	246	9	255	7
5/1000/0	256	13	243	16	324	12	336	10
5/100/1	228	11	214	10	232	8	242	6
5/500/1	372	10	358	12	410	9	415	7
5/1000/1	851	16	836	20	924	14	938	13
5/100/2	212	8	200	10	220	8	226	8
5/500/2	566	9	555	11	608	7	603	7
5/1000/2	1439	13	1426	21	1306	15	1513	13
10/100/0	199	9	184	9	200	9	210	6
10/500/0	223	10	210	13	259	9	269	7
10/1000/0	329	17	315	20	402	13	420	13
10/100/1	197	9	181	10	199	8	208	6
10/500/1	756	11	738	13	789	8	801	7
10/1000/1	1673	12	1656	16	1738	12	1749	9
10/100/2	183	7	174	8	189	7	196	4
10/500/2	1127	13	1118	10	1159	9	1174	9
10/1000/2	3466	15	3460	15	3527	11	3539	12
Overall	688	11	678	13	716	11	736	8

Table F.17. *Classification accuracy of ANN and SVM for datasets generated with fifth seed*

Problem Setting	**ANN (%)**			**SVM (%)**		
	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 1**	**Sensor 2**	**Sensor 3**
5/100/0	78	85	76	90	93	81
5/500/0	78	79	75	84	87	85
5/1000/0	78	83	76	82	93	80
5/100/1	78	85	42	90	93	24
5/500/1	78	79	38	84	87	30
5/1000/1	78	83	36	82	93	34
5/100/2	78	49	42	90	31	24
5/500/2	78	41	38	84	31	30
5/1000/2	78	37	36	82	33	34
10/100/0	76	81	76	72	93	73
10/500/0	89	72	83	95	79	86
10/1000/0	91	67	87	99	79	96
10/100/1	76	81	69	72	93	22
10/500/1	89	72	44	95	79	33
10/1000/1	91	67	38	99	79	35
10/100/2	76	58	69	72	31	22
10/500/2	89	38	44	95	34	33
10/1000/2	91	42	38	99	32	35
Overall	82	67	56	87	69	48

Table F.18. *Computational results for datasets generated with fifth seed*

Problem Setting	ECDST				ECY				ECMDF				ECMDFS			
	PCP	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**	**PCP**	**AC**	**CC**	**WC**
5/100/0	81.82	0.1749	0.2474	-0.1516	75.76	0.1156	0.2177	-0.2036	72.73	0.1334	0.2284	-0.1198	75.76	0.1460	0.2358	-0.1347
5/500/0	83.03	0.1383	0.1891	-0.1103	84.85	0.1468	0.2055	-0.1822	81.82	0.1208	0.1744	-0.1206	81.82	0.1238	0.1767	-0.1138
5/1000/0	81.21	0.1463	0.2149	-0.1504	77.27	0.1059	0.2051	-0.2316	78.18	0.1077	0.1786	-0.1464	78.79	0.1110	0.1822	-0.1534
5/100/1	81.82	0.2100	0.2743	-0.0794	72.73	0.1372	0.2302	-0.1110	81.82	0.1729	0.2270	-0.0705	84.85	0.2183	0.2820	-0.1385
5/500/1	85.45	0.2336	0.2980	-0.1446	81.82	0.1995	0.2683	-0.1105	85.45	0.2183	0.2768	-0.1256	84.85	0.2416	0.3101	-0.1416
5/1000/1	81.52	0.2371	0.3142	-0.1028	77.88	0.1799	0.2588	-0.0979	81.82	0.1914	0.2533	-0.0868	79.70	0.2249	0.3072	-0.0979
5/100/2	54.55	0.1238	0.2909	-0.0767	39.39	0.0105	0.2194	-0.1253	36.36	-0.0152	0.2130	-0.1456	51.52	0.0865	0.2473	-0.0844
5/500/2	78.79	0.2198	0.3020	-0.0855	35.15	-0.0872	0.0515	-0.1624	49.70	-0.0468	0.0404	-0.1330	80.00	0.1729	0.2393	-0.0924
5/1000/2	76.06	0.2149	0.2997	-0.0546	34.85	-0.0984	-0.0378	-0.1308	73.03	-0.0196	-0.0182	-0.0237	76.36	0.1809	0.2607	-0.0772
10/100/0	69.70	0.1482	0.2786	-0.1519	69.70	0.1167	0.2345	-0.1542	72.73	0.1373	0.2452	-0.1504	69.70	0.1301	0.2419	-0.1269
10/500/0	90.30	0.2197	0.2585	-0.1417	88.48	0.1907	0.2357	-0.1550	89.70	0.1924	0.2262	-0.1012	89.70	0.2020	0.2397	-0.1260
10/1000/0	90.30	0.1892	0.2257	-0.1509	89.70	0.1863	0.2265	-0.1637	89.39	0.1824	0.2192	-0.1274	89.09	0.1825	0.2206	-0.1283
10/100/1	60.61	0.1251	0.2835	-0.1185	60.61	0.1081	0.2709	-0.1423	66.67	0.1141	0.2199	-0.0974	63.64	0.1212	0.2410	-0.0883
10/500/1	87.88	0.2719	0.3234	-0.1012	74.55	0.1296	0.2284	-0.1598	78.79	0.1184	0.1752	-0.0925	83.64	0.2122	0.2718	-0.0926
10/1000/1	85.45	0.2549	0.3135	-0.0898	76.36	0.1597	0.2498	-0.1312	76.97	0.1224	0.1907	-0.1057	82.42	0.2253	0.2940	-0.0968
10/100/2	66.67	0.1471	0.2837	-0.1261	48.48	0.0296	0.2134	-0.1434	51.52	0.0397	0.1961	-0.1264	51.52	0.0686	0.2300	-0.1030
10/500/2	87.27	0.3029	0.3602	-0.0895	44.85	-0.0880	0.0481	-0.1986	61.82	-0.0335	0.0352	-0.1447	86.67	0.2397	0.2913	-0.0957
10/1000/2	86.06	0.2766	0.3366	-0.0932	37.58	-0.1211	-0.0141	-0.1855	62.42	-0.0435	0.0018	-0.1187	87.27	0.2484	0.3020	-0.1194
Overall	79.36	0.2019	0.2830	-0.1122	65.00	0.0790	0.1840	-0.1549	71.72	0.0940	0.1713	-0.1131	77.63	0.1742	0.2541	-0.1117

Table F.19. *Chosen scaling intervals for datasets generated with fifth seed*

Problem Setting	**ECMDF**	**ECMDFS**
5/100/0	[1, 2]	[1, 10]
5/500/0	[1, 5]	[1, 2]
5/1000/0	[1, 20]	[1, 10]
5/100/1	[1, 5]	[1, 2]
5/500/1	[1, 40]	[1, 20]
5/1000/1	[1, 40]	[1, 40]
5/100/2	[1, 2]	[1, 2]
5/500/2	[1, 2]	[1, 2]
5/1000/2	[1, 2]	[1, 2]
10/100/0	[1, 2]	[1, 2]
10/500/0	[1, 2]	[1, 40]
10/1000/0	[1, 5]	[1, 2]
10/100/1	[1, 2]	[1, 5]
10/500/1	[1, 10]	[1, 2]
10/1000/1	[1, 20]	[1, 40]
10/100/2	[1, 2]	[1, 2]
10/500/2	[1, 2]	[1, 2]
10/1000/2	[1, 40]	[1, 2]

Table F.20. *Computational times for datasets generated with fifth seed*

Problem Setting	ECDST		ECY		ECMDF		ECMDFS	
	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)	Training Time (sec)	Test Time (sec)
5/100/0	92	4	142	6	149	4	452	4
5/500/0	208	8	200	12	247	8	252	6
5/1000/0	251	13	240	17	320	12	331	10
5/100/1	182	7	174	8	186	7	194	4
5/500/1	378	10	366	13	420	9	422	7
5/1000/1	438	17	418	20	512	14	526	12
5/100/2	209	9	196	8	211	8	222	5
5/500/2	643	11	623	13	694	11	699	9
5/1000/2	660	12	648	16	506	10	726	10
10/100/0	202	9	188	9	204	8	214	5
10/500/0	228	10	215	13	261	8	270	6
10/1000/0	360	12	347	20	426	11	434	10
10/100/1	194	8	178	8	194	8	203	6
10/500/1	652	11	640	12	694	8	698	7
10/1000/1	2580	12	2543	18	2629	11	2650	9
10/100/2	101	4	215	7	220	6	226	5
10/500/2	1076	9	1057	13	1110	8	1120	7
10/1000/2	905	12	872	18	953	11	972	9
Overall	520	10	515	13	552	9	590	7

G. COMPUTATIONAL RESULTS FOR THRESHOLD VALUE FOR HYBRID MODELS

Table G.1. *Detailed results of ECMDFS in problem setting 10/100/1 for different levels of conflicting degree, k*

Problem Setting	Number of Correctly Predicted Targets		Number of DST Activation	Number of ECMDFS Activation	PCP (%)	
	h-ECMDFS	ECDST			h-ECMDFS	ECDST
10/100/1/0.75/1	25	25	32	1	75.76	75.76
10/100/1/0.80/1	25	25	33	0	75.76	75.76
10/100/1/0.85/1	25	25	33	0	75.76	75.76
10/100/1/0.90/1	25	25	33	0	75.76	75.76
10/100/1/0.95/1	25	25	33	0	75.76	75.76
10/100/1/1.00/1	25	25	33	0	75.76	75.76
10/100/1/0.75/2	22	22	33	0	66.67	66.67
10/100/1/0.80/2	22	22	33	0	66.67	66.67
10/100/1/0.85/2	22	22	33	0	66.67	66.67
10/100/1/0.90/2	22	22	33	0	66.67	66.67
10/100/1/0.95/2	22	22	33	0	66.67	66.67
10/100/1/1.00/2	22	22	33	0	66.67	66.67
10/100/1/0.75/3	25	25	33	0	75.76	75.76
10/100/1/0.80/3	25	25	33	0	75.76	75.76
10/100/1/0.85/3	25	25	33	0	75.76	75.76
10/100/1/0.90/3	25	25	33	0	75.76	75.76
10/100/1/0.95/3	25	25	33	0	75.76	75.76
10/100/1/1.00/3	25	25	33	0	75.76	75.76
10/100/1/0.75/4	28	27	29	4	84.85	81.82
10/100/1/0.80/4	28	27	29	4	84.85	81.82
10/100/1/0.85/4	28	27	30	3	84.85	81.82
10/100/1/0.90/4	28	27	30	3	84.85	81.82
10/100/1/0.95/4	28	27	31	2	84.85	81.82
10/100/1/1.00/4	27	27	33	0	81.82	81.82
10/100/1/0.75/5	20	20	30	3	60.61	60.61

Table G.1 (Continued)

10/100/1/0.80/5	20	20	32	1	60.61	60.61
10/100/1/0.85/5	20	20	33	0	60.61	60.61
10/100/1/0.90/5	20	20	33	0	60.61	60.61
10/100/1/0.95/5	20	20	33	0	60.61	60.61
10/100/1/1.00/5	20	20	33	0	60.61	60.61

Table G.2. *Detailed results of ECMDFS in problem setting 10/100/2 for different levels of conflicting degree, k*

Problem Setting	**Number of Correctly Predicted Targets**		**Number of DST Activation**	**Number of ECMDFS Activation**	**PCP (%)**	
	h-ECMDFS	**ECDST**			**h-ECMDFS**	**ECDST**
10/100/2/0.75/1	25	25	25	8	75.76	75.76
10/100/2/0.80/1	25	25	26	7	75.76	75.76
10/100/2/0.85/1	25	25	31	2	75.76	75.76
10/100/2/0.90/1	25	25	32	1	75.76	75.76
10/100/2/0.95/1	25	25	32	1	75.76	75.76
10/100/2/1.00/1	25	25	33	0	75.76	75.76
10/100/2/0.75/2	18	18	32	1	54.55	54.55
10/100/2/0.80/2	18	18	33	0	54.55	54.55
10/100/2/0.85/2	18	18	33	0	54.55	54.55
10/100/2/0.90/2	18	18	33	0	54.55	54.55
10/100/2/0.95/2	18	18	33	0	54.55	54.55
10/100/2/1.00/2	18	18	33	0	54.55	54.55
10/100/2/0.75/3	23	25	29	4	69.70	75.76
10/100/2/0.80/3	24	25	31	2	72.73	75.76
10/100/2/0.85/3	24	25	32	1	72.73	75.76
10/100/2/0.90/3	25	25	33	0	75.76	75.76
10/100/2/0.95/3	25	25	33	0	75.76	75.76
10/100/2/1.00/3	25	25	33	0	75.76	75.76
10/100/2/0.75/4	25	25	27	6	75.76	75.76
10/100/2/0.80/4	25	25	31	2	75.76	75.76

Table G.2 (Continued)

10/100/2/0.85/4	25	25	33	0	75.76	75.76
10/100/2/0.90/4	25	25	33	0	75.76	75.76
10/100/2/0.95/4	25	25	33	0	75.76	75.76
10/100/2/1.00/4	25	25	33	0	75.76	75.76
10/100/2/0.75/5	22	22	32	1	66.67	66.67
10/100/2/0.80/5	22	22	33	0	66.67	66.67
10/100/2/0.85/5	22	22	33	0	66.67	66.67
10/100/2/0.90/5	22	22	33	0	66.67	66.67
10/100/2/0.95/5	22	22	33	0	66.67	66.67
10/100/2/1.00/5	22	22	33	0	66.67	66.67

H. DETAILED EXPERIMENTAL RESULTS OF HYBRID MULTI-SENSOR TARGET CLASSIFICATION MODELS

Table H.1. *PCP values of h-ECMDF and h-ECMDFS for datasets generated with first seed*

Problem Setting	**h-ECMDF**				**h-ECMDFS**			
	PCP (%)	**AC**	**CC**	**WC**	**PCP (%)**	**AC**	**CC**	**WC**
5/100/0	87.88	0.1394	0.1745	-0.1151	87.88	0.1394	0.1745	-0.1151
5/500/0	91.52	0.0894	0.1159	-0.1970	91.52	0.0895	0.1160	-0.1970
5/1000/0	88.79	0.0942	0.1196	-0.1068	88.79	0.0946	0.1200	-0.1068
5/100/1	90.91	0.2563	0.2846	-0.0264	90.91	0.2563	0.2846	-0.0264
5/500/1	91.52	0.2408	0.2734	-0.1107	91.52	0.2424	0.2750	-0.1097
5/1000/1	86.06	0.2245	0.2835	-0.1396	86.06	0.2255	0.2839	-0.1353
5/100/2	81.82	0.2402	0.3287	-0.1581	84.85	0.2757	0.3207	0.0235
5/500/2	82.42	0.2575	0.3164	-0.0187	82.42	0.2575	0.3164	-0.0187
5/1000/2	79.39	0.2489	0.3438	-0.1167	79.39	0.2489	0.3438	-0.1167
10/100/0	72.73	0.0816	0.1691	-0.1518	72.73	0.0816	0.1691	-0.1518
10/500/0	89.09	0.1574	0.1970	-0.1657	90.30	0.1596	0.1961	-0.1807
10/1000/0	88.18	0.1262	0.1599	-0.1256	88.18	0.1260	0.1595	-0.1237
10/100/1	75.76	0.1856	0.3056	-0.1896	75.76	0.1856	0.3056	-0.1896
10/500/1	92.12	0.2418	0.2737	-0.1312	93.33	0.2437	0.2727	-0.1628
10/1000/1	89.09	0.2222	0.2652	-0.1283	89.09	0.2225	0.2657	-0.1299
10/100/2	75.76	0.2610	0.3491	-0.0146	75.76	0.2462	0.3658	-0.1278
10/500/2	75.76	0.2088	0.3206	-0.1405	80.61	0.2342	0.3107	-0.0836
10/1000/2	85.45	0.2968	0.3628	-0.0908	85.45	0.2968	0.3628	-0.0908
Overall	84.68	0.1985	0.2580	-0.1182	85.25	0.2014	0.2580	-0.1135

Table H.2. *Number of activation times of hybrid models and the classical DST for datasets generated with first seed*

Problem Setting	**Number of DST Used**	**Number of ECMDF/ECMDFS Used**
5/100/0	33	0
5/500/0	162	3
5/1000/0	322	8
5/100/1	33	0
5/500/1	162	3
5/1000/1	322	8
5/100/2	30	3
5/500/2	165	0
5/1000/2	330	0
10/100/0	33	0
10/500/0	159	6
10/1000/0	316	14
10/100/1	33	0
10/500/1	159	6
10/1000/1	316	14
10/100/2	26	7
10/500/2	151	14
10/1000/2	330	0

Table H.3. *PCP values of h-ECMDF and h-ECMDFS for datasets generated with second seed*

Problem Setting	h-ECMDF				h-ECMDFS			
	PCP (%)	**AC**	**CC**	**WC**	**PCP (%)**	**AC**	**CC**	**WC**
5/100/0	87.88	0.2120	0.2621	-0.1512	87.88	0.2135	0.2637	-0.1512
5/500/0	94.55	0.1431	0.1599	-0.1479	93.94	0.1425	0.1602	-0.1327
5/1000/0	91.82	0.1573	0.1834	-0.1363	91.82	0.1579	0.1843	-0.1378
5/100/1	81.82	0.2403	0.3102	-0.0744	81.82	0.2401	0.3100	-0.0744
5/500/1	90.91	0.2531	0.2831	-0.0474	90.91	0.2538	0.2846	-0.0535
5/1000/1	89.09	0.2545	0.2991	-0.1095	89.70	0.2572	0.3002	-0.1175
5/100/2	63.64	0.0939	0.2729	-0.2194	75.76	0.1883	0.2609	-0.0386
5/500/2	91.52	0.3222	0.3626	-0.1136	91.52	0.3222	0.3626	-0.1136
5/1000/2	83.64	0.2638	0.3235	-0.0413	83.64	0.2638	0.3235	-0.0413
10/100/0	78.79	0.1754	0.2315	-0.0330	78.79	0.1754	0.2315	-0.0330
10/500/0	84.85	0.1733	0.2234	-0.1072	84.85	0.1733	0.2234	-0.1072
10/1000/0	89.70	0.1372	0.1685	-0.1352	89.70	0.1367	0.1682	-0.1383
10/100/1	66.67	0.0690	0.1143	-0.0217	66.67	0.0690	0.1143	-0.0217
10/500/1	82.42	0.2584	0.3357	-0.1042	83.03	0.2624	0.3367	-0.1007
10/1000/1	82.12	0.2217	0.3010	-0.1428	84.24	0.2334	0.2993	-0.1192
10/100/2	54.55	0.0609	0.1374	-0.0308	54.55	0.0609	0.1374	-0.0308
10/500/2	83.03	0.2762	0.3484	-0.0773	83.03	0.2762	0.3484	-0.0773
10/1000/2	84.55	0.2932	0.3708	-0.1315	86.36	0.3023	0.3677	-0.1121
Overall	82.31	0.2003	0.2604	-0.1014	83.23	0.2071	0.2598	-0.0889

Table H.4. *Number of activation times of hybrid models and the classical DST for datasets generated with second seed*

Problem Setting	**Number of DST Used**	**Number of ECMDF/ECMDFS Used**
5/100/0	31	2
5/500/0	156	9
5/1000/0	319	11
5/100/1	31	2
5/500/1	156	9
5/1000/1	319	11
5/100/2	26	7
5/500/2	165	0
5/1000/2	330	0
10/100/0	33	0
10/500/0	163	2
10/1000/0	304	26
10/100/1	33	0
10/500/1	163	2
10/1000/1	304	26
10/100/2	33	0
10/500/2	165	0
10/1000/2	321	9

Table H.5. *PCP values of h-ECMDF and h-ECMDFS for datasets generated with third seed*

Problem Setting	h-ECMDF				h-ECMDFS			
	PCP (%)	**AC**	**CC**	**WC**	**PCP (%)**	**AC**	**CC**	**WC**
5/100/0	72.73	0.1572	0.2492	-0.0881	72.73	0.1572	0.2492	-0.0881
5/500/0	93.33	0.1452	0.1623	-0.0942	93.33	0.1532	0.1670	-0.0396
5/1000/0	86.97	0.1833	0.2385	-0.1855	86.97	0.1828	0.2378	-0.1842
5/100/1	72.73	0.1985	0.3122	-0.1047	72.73	0.1985	0.3122	-0.1047
5/500/1	89.70	0.2399	0.2814	-0.1213	93.94	0.2562	0.2784	-0.0875
5/1000/1	85.15	0.2478	0.3102	-0.1099	85.15	0.2480	0.3103	-0.1091
5/100/2	78.79	0.2080	0.2718	-0.0290	78.79	0.2080	0.2718	-0.0290
5/500/2	96.36	0.3926	0.4126	-0.1387	96.36	0.3926	0.4126	-0.1387
5/1000/2	83.64	0.2974	0.3679	-0.0633	83.64	0.2974	0.3679	-0.0633
10/100/0	78.79	0.1371	0.2024	-0.1053	78.79	0.1371	0.2024	-0.1053
10/500/0	89.70	0.1191	0.1469	-0.1227	89.70	0.1192	0.1470	-0.1220
10/1000/0	89.09	0.1123	0.1391	-0.1072	89.09	0.1126	0.1398	-0.1092
10/100/1	75.76	0.1964	0.2883	-0.0905	75.76	0.1964	0.2883	-0.0905
10/500/1	88.48	0.2366	0.2843	-0.1306	88.48	0.2367	0.2846	-0.1316
10/1000/1	86.67	0.2215	0.2727	-0.1112	86.67	0.2219	0.2733	-0.1122
10/100/2	75.76	0.1910	0.2856	-0.1048	72.73	0.1885	0.2886	-0.0783
10/500/2	82.42	0.2612	0.3341	-0.0808	83.03	0.2652	0.3338	-0.0709
10/1000/2	83.33	0.2625	0.3313	-0.0817	83.33	0.2625	0.3313	-0.0817
Overall	83.86	0.2115	0.2717	-0.1039	83.96	0.2130	0.2720	-0.0970

Table H.6. *Number of activation times of hybrid models and the classical DST for datasets generated with third seed*

Problem Setting	**Number of DST Used**	**Number of ECMDF/ECMDFS Used**
5/100/0	33	0
5/500/0	152	13
5/1000/0	327	3
5/100/1	33	0
5/500/1	152	13
5/1000/1	327	3
5/100/2	33	0
5/500/2	165	0
5/1000/2	330	0
10/100/0	33	0
10/500/0	162	3
10/1000/0	314	16
10/100/1	33	0
10/500/1	162	3
10/1000/1	314	16
10/100/2	31	2
10/500/2	164	1
10/1000/2	330	0

Table H.7. *PCP values of h-ECMDF and h-ECMDFS for datasets generated with fourth seed*

Problem Setting	h-ECMDF				h-ECMDFS			
	PCP (%)	**AC**	**CC**	**WC**	**PCP (%)**	**AC**	**CC**	**WC**
5/100/0	78.79	0.1643	0.2550	-0.1725	78.79	0.1643	0.2550	-0.1725
5/500/0	92.12	0.2100	0.2406	-0.1475	91.52	0.2097	0.2417	-0.1357
5/1000/0	95.45	0.1668	0.1802	-0.1142	95.76	0.1688	0.1823	-0.1361
5/100/1	75.76	0.1899	0.2847	-0.1064	75.76	0.1899	0.2847	-0.1064
5/500/1	83.03	0.2621	0.3332	-0.0859	82.42	0.2622	0.3358	-0.0833
5/1000/1	88.18	0.2687	0.3197	-0.1120	90.00	0.2800	0.3221	-0.0991
5/100/2	72.73	0.1414	0.2206	-0.0699	72.73	0.1414	0.2206	-0.0699
5/500/2	76.97	0.2357	0.3137	-0.0251	76.97	0.2357	0.3137	-0.0251
5/1000/2	89.39	0.3396	0.3861	-0.0524	89.39	0.3396	0.3861	-0.0524
10/100/0	81.82	0.1748	0.2358	-0.0994	84.85	0.1789	0.2336	-0.1273
10/500/0	89.09	0.1724	0.2129	-0.1587	89.09	0.1721	0.2128	-0.1605
10/1000/0	86.06	0.1329	0.1759	-0.1328	85.76	0.1324	0.1758	-0.1289
10/100/1	81.82	0.1929	0.2526	-0.0762	84.85	0.1989	0.2497	-0.0861
10/500/1	84.85	0.2545	0.3154	-0.0863	85.45	0.2551	0.3145	-0.0938
10/1000/1	84.24	0.2190	0.2842	-0.1297	84.55	0.2191	0.2834	-0.1325
10/100/2	69.70	0.2144	0.3751	-0.1553	75.76	0.2395	0.3546	-0.1202
10/500/2	71.52	0.1631	0.2475	-0.0486	71.52	0.1631	0.2475	-0.0486
10/1000/2	77.88	0.2687	0.3701	-0.0880	77.88	0.2687	0.3701	-0.0880
Overall	82.19	0.2095	0.2780	-0.1034	82.95	0.2122	0.2769	-0.1037

Table H.8. *Number of activation times of hybrid models and the classical DST for datasets generated with fourth seed*

Problem Setting	**Number of DST Used**	**Number of ECMDF/ECMDFS Used**
5/100/0	33	0
5/500/0	161	4
5/1000/0	304	26
5/100/1	33	0
5/500/1	161	4
5/1000/1	304	26
5/100/2	33	0
5/500/2	165	0
5/1000/2	330	0
10/100/0	29	4
10/500/0	158	7
10/1000/0	311	19
10/100/1	29	4
10/500/1	158	7
10/1000/1	311	19
10/100/2	31	2
10/500/2	165	0
10/1000/2	330	0

Table H.9. *PCP values of h-ECMDF and h-ECMDFS for datasets generated with fifth seed*

Problem Setting	h-ECMDF				h-ECMDFS			
	PCP (%)	**AC**	**CC**	**WC**	**PCP (%)**	**AC**	**CC**	**WC**
5/100/0	78.79	0.1629	0.2444	-0.1396	78.79	0.1628	0.2444	-0.1403
5/500/0	83.64	0.1400	0.1914	-0.1223	83.64	0.1400	0.1911	-0.1213
5/1000/0	80.30	0.1360	0.2068	-0.1530	80.30	0.1352	0.2070	-0.1578
5/100/1	81.82	0.2056	0.2690	-0.0794	81.82	0.2052	0.2685	-0.0794
5/500/1	85.45	0.2317	0.2944	-0.1368	84.85	0.2321	0.2967	-0.1298
5/1000/1	81.21	0.2281	0.3034	-0.0975	80.61	0.2283	0.3062	-0.0954
5/100/2	45.45	0.0834	0.2965	-0.0941	54.55	0.1162	0.2769	-0.0767
5/500/2	76.97	0.2103	0.3013	-0.0940	78.79	0.2171	0.2990	-0.0870
5/1000/2	76.06	0.2149	0.2997	-0.0546	76.06	0.2149	0.2997	-0.0546
10/100/0	69.70	0.1483	0.2788	-0.1519	69.70	0.1466	0.2763	-0.1519
10/500/0	90.30	0.2192	0.2579	-0.1417	90.30	0.2195	0.2582	-0.1417
10/1000/0	90.00	0.1901	0.2270	-0.1422	90.30	0.1902	0.2267	-0.1499
10/100/1	60.61	0.1231	0.2802	-0.1185	60.61	0.1184	0.2724	-0.1185
10/500/1	86.67	0.2656	0.3230	-0.1080	87.27	0.2673	0.3211	-0.1017
10/1000/1	80.91	0.2241	0.3038	-0.1138	82.73	0.2349	0.3021	-0.0870
10/100/2	66.67	0.1471	0.2837	-0.1261	66.67	0.1471	0.2837	-0.1261
10/500/2	86.06	0.2925	0.3578	-0.1108	87.27	0.2988	0.3554	-0.0895
10/1000/2	84.85	0.2668	0.3356	-0.1181	86.06	0.2739	0.3342	-0.0985
Overall	78.08	0.1939	0.2808	-0.1168	78.91	0.1971	0.2789	-0.1115

Table H.60. *Number of activation times of hybrid models and the classical DST for datasets generated with fifth seed*

Problem Setting	**Number of DST Used**	**Number of ECMDF/ECMDFS Used**
5/100/0	32	1
5/500/0	158	7
5/1000/0	303	27
5/100/1	32	1
5/500/1	158	7
5/1000/1	303	27
5/100/2	30	3
5/500/2	161	4
5/1000/2	330	0
10/100/0	32	1
10/500/0	163	2
10/1000/0	304	26
10/100/1	32	1
10/500/1	163	2
10/1000/1	304	26
10/100/2	33	0
10/500/2	163	2
10/1000/2	324	6