DECISION FUSION FOR SUPERVISED, UNSUPERVISED AND SEMI-SUPERVISED LEARNING

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

BY

METE ÖZAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JUNE 2013

Approval of the thesis:

DECISION FUSION FOR SUPERVISED, UNSUPERVISED AND SEMI-SUPERVISED LEARNING

submitted by **METE ÖZAY** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Fatoş T. Yarman Vural
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Göktürk Üçoluk
Computer Engineering Department, METU _____

Prof. Dr. Fatoş T. Yarman Vural
Computer Engineering Department, METU _____

Prof. Dr. Adnan Yazıcı
Computer Engineering Department, METU _____

Assist. Prof. Dr. Sinan Kalkan
Computer Engineering Department, METU _____

Assist. Prof. Dr. Pınar Duygulu Şahin
Computer Engineering Department, Bilkent University _____

Date: _____

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

Name, Last Name: METE ÖZAY

Signature :

ABSTRACT

DECISION FUSION FOR SUPERVISED, UNSUPERVISED AND SEMI-SUPERVISED LEARNING

Özay, Mete

Ph.D., Department of Computer Engineering

Supervisor : Prof. Dr. Fatoş T. Yarman Vural

June 2013, 176 pages

In this thesis, Decision Fusion approaches have been analyzed for Supervised, Unsupervised and Semi-supervised Learning problems. In Supervised Learning, classification or generalization error minimization problem has been studied by analyzing the classification error of a classification algorithm into two parts. In the first part, the minimization of the difference between N-sample and large-sample classification error of k-NN has been studied using a hierarchical decision fusion algorithm called Fuzzy Stacked Generalization (FSG). Then, a weighted decision fusion and two sample selection algorithms are proposed to minimize the difference between large-sample error and Bayes Error in FSG.

Unsupervised image segmentation problem has been analyzed for the fusion of decisions of different segmentation algorithms. An unsupervised decision fusion algorithm called Segmentation Fusion (SF) is proposed together with two distance learning methods. In addition, a weighted decision fusion method has been introduced. Two algorithms are suggested for the estimation of algorithm parameters and the number of different segmentation labels. The prior and side information about the statistical properties of data are integrated to SF using a new decision fusion algorithm called Semi-supervised Segmentation Fusion. The proposed algorithms and methods have been analyzed and examined on both synthetic and real-world datasets.

Keywords: Data Fusion, Statistical Learning, Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Classification, Segmentation

ÖZ

DENETİMLİ, DENETİMSİZ VE YARI-DENETİMLİ ÖĞRENME İÇİN VERİ FÜZYONU

Özay, Mete

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Fatoş T. Yarman Vural

Haziran 2013 , 176 sayfa

Bu tez çalışmasında, Denetimli, Denetimsiz ve Yarı-denetimli Öğrenme problemleri için veri füzyonu yaklaşımları incelenmiştir. Denetimli Öğrenme için, sınıflandırma veya genelleştirme hatasını azaltma problemi, bir sınıflandırma algoritmasının sınıflandırma hatasının iki kısımda analiz edilmesiyle incelenmiştir. İlk kısımda, k-NN algoritmasının N sayıda örnek ve çok sayıda örnek kullanılarak elde edilen sınıflandırma hataları arasındaki farkın azaltılması, Bulanık Yığılmış Genelleme Algoritması olarak adlandırılan bir hiyerarşik sınıflandırma algoritması kullanılarak sağlanmıştır. Daha sonra, çok sayıda örnek kullanılarak hesaplanan sınıflandırma hatası ve Bayes Hatası arasındaki farkı Bulanık Yığılmış Genelleme algoritmasında azaltmak amacıyla, bir ağırlıklı karar füzyonu ve iki örnek seçim algoritması sunulmuştur.

Denetimsiz görüntü bölütlenmesi problemi, farklı bölütleme algoritma kararlarının füzyonu için incelenmiştir. Bölütleme Füzyonu (BF) olarak adlandırılan bir karar füzyonu ile birlikte iki uzaklık öğrenme metodu önerilmiştir. Ayrıca, bir ağırlıklı karar füzyonu metodu sunulmuştur. Algoritma parametrelerinin ve farklı bölütleme etiketlerinin sayısının kestirimi için iki algoritma önerilmiştir. Yarı-denetimli Bölütleme Füzyonu olarak adlandırılan yeni bir karar füzyonu algoritması kullanılarak BF algoritmasına verilerin istatistiksel özellikleri hakkında ön bilgiler ve yan bilgiler entegre edilmiştir. Önerilen algoritmalar ve metodlar, hem sentetik hem de gerçek veri kümeleri üzerinde analiz edilmiş ve incelenmiştir.

Anahtar Kelimeler: Veri Füzyonu, İstatistiksel Öğrenme, Denetimli Öğrenme, Denetimsiz Öğrenme, Yarı-denetimli Öğrenme, Sınıflandırma, Bölütleme

To $\tau\chi$ and My Family

ACKNOWLEDGMENTS

This work has been started and completed by the efforts and support of Professor Fatos Tunay Yarman Vural. She has been the fountain, the core, and the soul of this work. Without her efforts, neither this work could be started nor completed. In addition to her support to the scientific progress, she has supported philosophical, psychological, sociological and spiritual development of the work, which cannot be disentangled from the scientific part. Although we have always disagreements about the ideas, we succeeded to convert the disagreements to new products, as observed in the evolution structure of the nature. I hope that this productive interactive will be life-long and besides. I am grateful to Professor H. Vincent Poor and Professor Sanjeev R. Kulkarni from Princeton University. Besides giving me a chance to study with them in Princeton University, their comments and suggestions on my works have enlightened my view for scientific production, life and culture. I would like to thank to Professor Sinan Kalkan and Professor Ales Leonardis, who have enabled this work to be completed earlier than the expected time. I would like to thank to Professor Pinar Duygulu Sahin for her helpful comments and suggestions on the topics in the thesis. I thank to all my friends in METU Image lab, specially, Dr. Caglar Senaras, Dr. Sarper Alkan, Dr. Gulsah Tumuklu, Orhan Firat, Itir Onal, Omer Ekmekci and Okan Akalin for their support and comments during my studies. I thank to all my officemates and colleagues in Informatics Institute, especially, Berna Bakir, Selim Nar, Professsor Alptekin Temizel and Professor Tugba Taskaya Temizel. I would like to thank to Dr. Haipeng Zheng and Dr. Inaki Esnaola from Princeton University for their friendship and collaboration. I thank to Serkan Polad, who has been my life-long friend and colleague, and a comrade during my journey in science. I thank to Professor Huseyin Vural for his humanitarian efforts not only for my progress but also for the development of the *little* scientists of the world. Finally, I am grateful to my mother, my father and my brothers for their endless support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

CLT	Computational Learning Theory
SLT	Statistical Learning Theory
ALT	Algorithmic Learning Theory
FSG	Fuzzy Stacked Generalization
k-NN	k Nearest Neighbors
DM	Decision Margin
DFDM	Decision Fusion using Decision Margin
SS-1	Spurious training data elimination from meta-layer input training dataset
SS-2	Spurious training data elimination from base-layer input training dataset
BOEM	Best One Element Move
BOK	Best of K
SF	Segmentation Fusion
SDD	Symmetric Distance Function
Average SOD	Average Sum of Distances
SSSF	Semi-supervised Segmentation Fusion
ADMM	Alternating Direction Method of Multipliers
SW	Segmentation Weighting
DL	Distance Learning
QD	Quasi-distance Learning algorithms
RHSEG	Recursive Hierarchical Segmentation Algorithm
RI	Rand Index
ARI	Adjusted Rand Index
HI	Hubert's Index
MI	Mirkin's Index
AMI	Adjusted Mutual Information
$\mathcal{MI}$	Mutual Information
$\mathbb{H}$	Entropy

CHAPTER 1

INTRODUCTION

1.1 Existence of Knowledge of Beings

In nature, all beings are faced with a spectrum of problems from existence to non-existence. As human beings, one of our concerns has been solving our problems for the sake of understanding the nature or understanding the nature for the sake of solving our problems. In order to understand the mechanism of nature and solve the problems with maximum efficiency and minimum effort, we systematized several methods in an intellectual way, called scientific methodology.

Scientific methods have influenced and have been inspired by Epistemology and Ontology which are concerned with the nature of knowledge and understanding, and being and existence, respectively. Although two individual philosophical studies seem different from each other and the scientific method, they are related with simple and fundamental questions, such as "*Does a solution of the problem exist?*", "*Do we have sufficient amount of data obtained from the observations on the beings to solve the problem?*", or "*Is our knowledge of the problem and the method complete?*", etc. Therefore, the passion of existence has been regenerated as the passion of knowledge to satisfy the ambition of beings.

These questions have been studied by the scientific method [1] constructing a collection of procedures used for pursuit of knowledge as

1. the formulation of a problem,
2. the collection of data through observation on beings (i.e. observables),
3. proposing explanations (hypotheses) for understanding and solving the problem using the data and
4. testing the explanations.

Initialized by the observation of a phenomenon, a scientific method implements these four steps until a problem is solved. If a method cannot solve the problem, three approaches are employed. First, the steps are implemented in a cycle, independently or recursively, until a solution is obtained. Second, the problem is divided into sub-problems, new methods are proposed to obtain solutions to sub-problems, and the solutions are merged to solve the problem. Third, the methods which are used to solve other problems than the problem of interest are used together with the proposed method. Then, interactions between the steps

and among different scientific methods during the iterations lead to new scientific methods in the hope of getting closer to the problem solution. Therefore, a scientific method can be considered as a self-organizing and dynamic system [2].

1.2 Seeking the Knowledge of Beings, Relationships and Explanations

One of the implications of the realization of the scientific method in computer science is algorithm [3]. Algorithms have been employed for formalizing and solving problems regarding the computation of hypotheses using the representation of knowledge of beings, their relations and explanations of the phenomena, i.e. the computation itself. Therefore, the above mentioned scientific approaches and implementation steps have been employed by algorithms in computer science for the passion of knowledge of beings.

In computer science, several ontological properties have been used to represent knowledge. For instance,

- samples are used to represent the knowledge of beings,
- attributes and features are used to extract information from the observations which represent the knowledge of samples,
- classes are used to represent the categorization of beings into groups using features, and
- algorithms are used to represent the explanations for the problems exposing relationships between these representations.

In general words, algorithms have been used to generate knowledge of beings and their relationships using information obtained from features. This *fascinating* and *beautiful* relationship between *information* and *computation* has been one of the motivations of this work.

1.3 Learning How to Group and Categorize Beings

The major goal of this work is to analyze one of the fundamental ontological problems of computer science, which is division of a set of beings into groups and their categorization. From the machine learning perspective, these problems have been referred as clustering and classification of samples, measured by a variety of data acquisition devices.

In the proposed algorithms, four steps of a scientific method have been employed using three approaches mentioned above to solve an epistemological problem, which is *learning*, i.e. generating (e.g. predicting or estimating) the knowledge of the relationships between samples, their categories and groups.

Three problems are considered according to the information about the *target* or *essential* categories and groups, available to the algorithms. These problems are called Supervised Learning, Unsupervised Learning and Semi-supervised Learning in Statistical Learning Theory [4, 5, 6]. In Unsupervised Learning, information about the samples is available through the observations, such as features extracted from visual and audio measurements. In Supervised Learning, additional information of samples about the target categories and groups is available.

Finally, in Semi-supervised Learning, information about the samples and about few target labels may be available, but instead of label information about most of samples, some external sources, such as expert knowledge which defines the relationship between the samples and their target values may be available. An overview of statistical learning algorithms which motivates this work is provided in Chapter 2.

Generally speaking, performance of a statistical learning algorithm is measured by the *similarity* between the target and the predicted values of the samples. In Statistical Learning Theory, the performance of an algorithm is estimated by computing the expected value of the *similarity* values, where the expectation is computed over the density of features in a feature space. If the size of the set of training samples is finite, then an *empirical* performance, such as the average performance, is measured over the samples in the dataset [7].

1.3.1 Learning with Fusion of Multiple Algorithms

Several statistical learning algorithms can be combined at different steps of a learning method in order to obtain better performance than that of a single statistical learning algorithm. These levels can be *i*) data acquisition, *ii*) feature extraction from data, and *iii*) making decision using features. The combination process is called *Fusion* and the fusion approaches are referred in general as *Data Fusion* [8, 9, 10, 11]. Then, Data Fusion approaches used at these levels are called *i*) *Data Level Fusion*, *ii*) *Feature Level Fusion*, and *iii*) *Decision Level Fusion* [8], respectively. Data Fusion approaches are summarized, and their relationship with Statistical Learning methods are given in Chapter 2.

Given a set of features which construct feature spaces, the classification problem can be described as computing hypotheses which *analyze* and *explain* the relationships between the features of the samples and their target values using classification algorithms. Once the hypotheses are computed, they are tested by predicting the targets of new samples, called test samples. For instance, in the well-known k Nearest Neighbors (k-NN) algorithm [12], the relationship between a test sample and the other samples in the dataset is defined as the belongingness of the samples to a set of k nearest *neighbors* of the test sample, which is computed using a *distance function* in a feature space consisting of features of observations on samples. Then, the target value of the test sample is predicted using several decision combination rules, such as voting, on target values of its k nearest neighbors.

1.3.2 The Concept of Error in Machine Learning

When an algorithm misclassifies a sample, an error is measured for that algorithm. Under some conditions, errors of classification algorithms may approach to the minimum achievable classification error rate, called *Bayes Error* (or *Bayes Risk*), over all the classification hypotheses. If the hypotheses do not satisfy these conditions, then a gap of error between their error values and that of Bayes occurs. In order to satisfy the ambition of achieving minimum error of the hypotheses, the algorithms which generate the hypotheses should be revised to get closer to the Bayes Error.

For instance, the classification error of k-NN approaches to the Bayes Error if the number of samples N approaches to infinity (i.e. $N \to \infty$) and the neighborhood size k increases with increasing sample size such that $\lim_{N\to\infty} \frac{k}{N} \to 0$ [13]. If these two conditions are not satisfied, then

a non-zero error difference between errors of classification algorithms employed on finite ($N << \infty$) number of samples and large ($N \to \infty$) number of samples is observed [13]. Classification error computed in the former case is called *N-sample* error, and in the latter case is called *large-sample* error. In order to minimize the error difference between N-sample and large-sample error, the relationship between the samples is redefined by designing a new distance function employed in k-NN [14]. Error minimization problem for classification is described in detail in Chapter 3.

Note that distance functions of k-NN algorithms are required to be updated in order to increase the *capability of information extraction* from the feature spaces to obtain more accurate decisions. If feature spaces provide discriminative information about samples, then a single classification rule applied on this feature space, such as a Naive Bayes or a linear classification rule, can provide high classification performances.

On the other hand, if a feature space does not provide *sufficient* information for the classification of the samples, then this feature space is mapped to a more informative feature space using linear or nonlinear mappings. This approach has motivated us to define distance function learning problem as a feature space mapping and a decision fusion problem.

1.4 Improvement of Learning Performance using Decision Fusion

The major theme of this study is to analyze the error bounds to improve the learning performance under a decision fusion framework. The thesis is arranged in two parts: In Part I, a distance learning approach used to minimize the difference between N-sample and large-sample error is employed in a decision fusion algorithm called Fuzzy Stacked Generalization (FSG). In addition, a weighted decision fusion algorithm and two sample selection algorithms are proposed to minimize large-sample error using FSG. Part II is devoted to a set of semi-supervised and unsupervised fusion algorithms. The approaches and contributions in Part I and II are summarized below.

1.4.1 Part I: Decision Fusion for Supervised Learning

This part of the thesis involves two chapters. A distance learning approach for classification error minimization and the proposed algorithms are given with experimental analyses in Chapter 3. Chapter 4 is devoted to the applications of the suggested decision fusion algorithms on the real-world problems, namely remote sensing and scene classification.

In Chapter 3, first several properties of feature spaces which provide sufficient information for designing distance functions are analyzed. Feature space mappings are employed in two steps. In the first step, k-NN algorithms are used to map feature spaces of the features of observations to *Decision Spaces* consisting of decision vectors that represent class membership predictions of k-NN algorithms on the samples. In the second step, decision spaces are fused in a new space called *Fusion Space* in order to employ ℓ_2 norm Euclidean distance as a distance function for classification. These steps are implemented in a hierarchical classification algorithm called Fuzzy Stacked Generalization (FSG). The description and analyses of FSG is provided in Chapter 3. In FSG, base-layer k-NN classifiers are used to map feature vectors of observations to decision vectors, which represent class memberships of samples, in decision spaces. At the

meta-layer, decision spaces are concatenated to construct a fusion space, and a meta-layer classification is employed for final prediction of class labels.

Once the gap between N-sample and large-sample error is bridged, the second goal is to achieve the Bayes Error, i.e. bridging the gap between large-sample error and Bayes Error in a classifier. Cover and Hart [13] state that this task can be achieved if and only if (iff) target class labels of samples are predicted with maximum *certainty* or *uncertainty*, i.e. iff the probability of assigning a sample to the target class is either 0, 1 (*with certainty*) or $\frac{1}{2}$ (*with uncertainty*) for each sample. This result is quite intuitive in a two-class classification problem using $k = 1$ in k-NN algorithm. In this particular example, expected error of assigning a sample to wrong classes is equal to the Bayes Error as shown in [13].

In order to satisfy the conditions stated by Cover and Hart [13], one must assure that the samples belonging to the same class reside in the same Voronoi region in the fusion space. Therefore, one needs to measure the contribution of each sample to the classification performance of the meta-layer classifier depending on its location in the feature space. For this purpose, a distance function called Decision Margin is proposed in Chapter 3. It is shown that if decision margin of the sample is minimum, then the feature vector of the sample is closer to the vertex which represents its target class label than the other vertices of a polytope of feature vectors in the fusion space. If decision margin values of all of the samples are minimum, then the samples are correctly classified with maximum certainty by the meta-layer classifier.

In order to minimize decision margin values of the samples in the fusion space, decisions of base-layer classifiers are fused with weights. This task is achieved by computing the weights which minimize decision margin under a convex optimization problem. In order to employ a decision selection strategy, which assigns zero weights to decisions of base-layer classifiers that provide decisions with *large* decision margin values, sparsity constraints are used in the optimization problem. Then, the optimization problem is solved using an optimization algorithm called Alternating Direction Method of Multipliers. A weighted decision fusion algorithm which minimizes decision margin values of samples by computing the weights using the training dataset is introduced in Chapter 3.

Note that the above mentioned conditions, which are required to achieve the Bayes Error, should be valid almost everywhere in the feature space, i.e. for almost every feature vector of each sample. An approach to assure its validity for each sample is the employment of sample selection methods considering the features of the samples in base-layer feature spaces and decision spaces. Basically, sample selection methods involve re-designing the training dataset by eliminating some of the samples which are assumed to decrease the large-sample error. In order to reduce the large-sample error, two sample selection methods are proposed for FSG. In the proposed selection methods, first the samples which could not be classified by at least one base-layer classifier are selected to construct a set of misclassified samples. Then, base-layer feature vectors and decision vectors of the samples belonging to the misclassified sample set are eliminated in the first and second sample selection algorithms, respectively. The details of the algorithms are given in Chapter 3.

In Chapter 4, the proposed algorithms have been employed to solve two practical problems, namely, target detection of remotely sensed objects using multi-spectral images and object classification in a scene by using audio-visual data. In the first application, a building detection problem is defined as a two class classification problem in remotely sensed images, where each segment of image pixels belongs to either building or background classes. Datasets contain

multi-spectral images which are obtained from a high-resolution satellite, called QuickBird [15]. In the second application, moving target detection problem is defined as a multi-class classification problem. In this application, datasets contain audio-visual measurements on the moving targets and other objects in an environment. Samples are labeled in four classes according to the cases where *i*) the first target, *ii*) the second target, *iii*) both the first and the second targets, and *iv*) none of the targets take place in an environment. Then, the four class classification problem is solved using the proposed algorithms.

1.4.2 Part II: Decision Fusion for Unsupervised and Semi-supervised Learning

In Supervised Learning, samples are provided by both observations represented in feature spaces and information about their target values such as class labels. When target class labels are represented as the members of a set of numbers in the space of integer numbers $\mathbb{Z}$ and observations are represented as the members of a set of numbers in the space of real numbers $\mathbb{R}$, a classification problem can be defined as computing a mapping between these two sets. Then the error function of the problem is a function of *difference* between predicted class labels and target class labels.

On the other hand, if target values are not given, then a fundamental epistemological and ontological problem occurs. From epistemological point of view, the information about the target values, such as cluster labels, is not available. Therefore, information about the co-occurrence of two samples in the same cluster is not available. From ontological point of view, existence of the target values is not assured. In other words, there may not exist a cluster in which two samples must or cannot co-occur.

One approach to provide an approximate solution to these problems is transforming the problem of modeling the relationships between the samples and their target values to the problem of modeling the relationships among the samples, called clustering, which is an ill-posed problem [4].

In data clustering problems, spatial relationships among feature vectors of samples may not be *physically meaningful*, therefore defining the target relationships and clusters may be a challenge. For instance, Kernel clustering algorithms [16] first transform feature spaces of feature vectors of samples to higher dimensional (usually infinite dimensional) spaces, and then cluster the features in these spaces. Therefore, target clusters and spatial relationships among the features cannot be defined in infinite dimensional spaces.

However, target clusters can be determined in some specific applications, such as image segmentation, in which pixels of an image are grouped into clusters called segments. In an image segmentation problem, target segments are labeled by users according to either human object perception, or physical properties, such as geophysical characteristics, of the objects. The former labeling procedure is employed in the segmentation of images consisting of objects in natural scenes, such as images collected from personal cameras. For instance, an image of a bird (with id number 42049) in Berkeley Image Segmentation Dataset [17] has been segmented into five different number of segments by five different users.

On the other hand, geophysical characteristics of objects, which will be represented by segments, are used in the determination of target segments of most of remotely sensed images. For instance, location of a building in the physical environment is physically recorded by

users. Then, labels of the pixels which represent that building in an image are determined using the records. Unsupervised and Semi-supervised Learning methods have been employed for segmentation of remotely sensed images in this work.

Although a segment, which represents the building pixels, is roughly labeled by expert users using *formally defined quantitative records*, determination of the boundary of a segment is still a challenge since the boundary is usually defined by humans. For instance, a tag of a region which represents a building can be obtained from the records. However, corners and edges of the roof of the building, which define the boundary of the building region, should be determined by users. Since each user may provide different segment boundaries, a *decision fusion* problem occurs in the combination of the decisions of users on a target segmentation.

A similar problem is observed for the fusion of segmentation decisions of different segmentation algorithms or that of a segmentation algorithm employed on an image with different parameters. If segmentation decisions of the *base-layer* segmentation algorithms are different from each other, then simple decision fusion methods such as voting cannot be applied. In this case, an approach used to solve the fusion problem is to make a *consensus* among the segmentation algorithms. A consensus error among segmentation decisions is defined by a *distance function* which measures the *difference* or *similarity* between segmentations [18]. For instance, a symmetric distance computed between different pairwise segmentations gives information about the number of pixels which co-occur in the same segment in different segmentations [18].

Computing the exact solution of the consensus segmentation problem is an NP-complete problem [18]. However, approximate solutions to the problem can be computed in polynomial time [18]. In order to solve the problem with less computational complexity, i.e. in linear time, stochastic optimization methods are employed in a new consensus segmentation algorithm called Segmentation Fusion (SF). Details of segmentation fusion problem and the suggested algorithm are given in Chapter 5.

Five challenges of segmentation fusion have been addressed in this work. The first challenge is to suggest a method for learning the *structure*, i.e. type of mapping employed in the distance function, using prior information about the data such as the distribution of pixels in an image. For instance, a consensus clustering algorithm called Best One Element Move (BOEM) [18] measures the consensus error between two segmentations using the Rand Index (RI) which provides information about the probability of agreement among the segmentations [19]. In order to minimize the consensus error defined by RI, a normalized symmetric distance function called Average Sum of Distances (SOD) is used [18]. SOD assumes that the distributions are uniform and the probability of the disagreement between two different segmentations for co-segmenting two elements is a Normal distribution. However, this may not be a valid assumption in real-world datasets as analyzed in Section 5.3. Therefore, the mappings of distance functions is redesigned, considering the statistical properties of the data.

For this purpose, two distance function learning methods called *Distance Learning* (DL) and *Quasi-distance Learning* (QD) are proposed in Section 5.6 and Section 5.7. In DL, we relax the equal segmentation size assumption of SF, which assumes that each segmentation obtained from the base-layer segmentation algorithms contains equal number of segments. DL computes an adjusted form of RI called Adjusted Rand Index (ARI) for each pair of segmentations with different number of segments, and defines the consensus error function as a function of ARI. However, ARI and DL assume that the number of pixels in each segment is the same. In order to relax this assumption, QD computes the distance function between two segmentations as

a function of conditional entropies of the segmentations. Therefore, QD does not make any assumption about distribution of pixels and segments in the dataset, which is taken into account for each different segmentation during the computation of the distance function.

The second challenge is the estimation of an algorithm parameter which controls the learning capacity, convergence rate and performance of the SF algorithm. The third challenge is to estimate the *optimal* number of different segment names or labels, which is used to obtain segmentations from the base-layer segmentation algorithms that lead to minimum distance function value to achieve a consensus. For the parameter and cluster label estimation problems, two performance indices which measure the consensus performance of the SF for the selected parameters, are defined. Then, the parameters which maximize the consensus performance are selected. Details of the algorithms are given in Chapter 5.

The fourth challenge is weighting the outputs of segmentation algorithms to achieve a better consensus before fusing them. In this approach, distance function learning is employed by learning the weight parameters of a weighted distance function, where a weight is assigned to each segmentation. The weight of a segmentation s_i is computed using ARI between s_i and other base-layer segmentations. Therefore, the weights are computed by minimizing a distance function which is defined as a function of ARI between segmentations, such as SOD. Details of the algorithm are given in Section 5.8.

Finally, integration of side information, such as the constraints that give information about the co-occurrence of pixels in the same and different segments, to the Segmentation Fusion algorithm has been studied. For this purpose, *must-link* and *cannot-link* constraints that provide side information is used to design the distance function of SF. Then, the distance function is enhanced as a weighted distance function of segmentation weights as mentioned above and in Section 5.8. In order to compute the weights using the constraints, segmentation fusion problem is defined as a convex optimization problem. An algorithm called Semi-supervised Segmentation Fusion (SSSF) which solves the optimization problem is given in Section 5.9.

Computational complexity analyses of FSG, SF and SSF are given in Section 3.6 and Section 5.10, respectively. Chapter 7 concludes the thesis.

1.5 Summary of the Chapters

In summary, Decision Fusion approaches to solve Supervised, Unsupervised and Semi-supervised Learning problems have been studied in this thesis. Specifically, pattern classification problem has been addressed in Supervised Learning and image segmentation problem has been analyzed in Unsupervised and Semi-supervised Learning.

The chapters of the thesis can be summarized as follows;

- Part I: Decision Fusion for Supervised Learning
  - **Chapter 2** gives the motivation of the thesis by criticizing the statistical learning methods and data fusion approaches studied in the literature.
  - **Chapter 3** analyzes classification error minimization problem using a distance learning approach, which is employed in a decision fusion algorithm called Fuzzy

Stacked Generalization (FSG). In addition, a weighted decision fusion and two sample selection algorithms are introduced to decrease the large-sample classification error. The proposed algorithms are examined in synthetic and real-world benchmark datasets.

  - **Chapter 4** employs the suggested decision fusion methods to solve two practical problems, namely, target detection of remotely sensed objects using multi-spectral images and object classification in a scene by using audio-visual data.

- Part II: Decision Fusion for Unsupervised and Semi-supervised Learning

  - **Chapter 5** introduces an unsupervised segmentation fusion algorithm which combines the segmentation outputs (i.e. decisions) of various segmentation algorithms employed on an image to achieve a consensus among the decisions. Then, the challenges of the unsupervised segmentation fusion algorithm proposed are experimentally and theoretically analyzed. In order to solve these challenges, various methods have been suggested by incorporating the prior information on the training datasets in the proposed segmentation fusion algorithm. In addition, the side information obtained from the expert knowledge has been used in a new Semi-supervised Segmentation Fusion algorithm.
  - **Chapter 6** provides the experimental analyses of the proposed unsupervised and semi-supervised segmentation fusion algorithms on benchmark aerial and multi-spectral image datasets.

- **Chapter 7** summarizes and discusses the contributions, the analyses and the results provided in the thesis.

CHAPTER 2

OUR MOTIVATION BASED ON A CRITIQUE OF STATISTICAL LEARNING METHODS AND DATA FUSION APPROACHES

In this chapter, we present our motivation and rationale for the suggested Supervised, Unsupervised and Semi-supervised learning methods given in Part I and Part II of this thesis. For this purpose, an overview together with a critique of the available Statistical Learning Methods is provided. Then, the major problems in Data Fusion Approaches are discussed. Since the chapter aims at a broad overview for the motivation of the proposed algorithms, mathematical formalisms are avoided and postponed until the subsequent chapters. The necessary background and literature survey are provided in Part I and II along with the suggested methods throughout the thesis.

2.1 An Overview of Statistical Learning Methods

Statistical Learning Theory (SLT) is a subfield of Computational Learning Theory (CLT) which deals with the analysis of the data and the functions employed on that data. Unlike Algorithmic Learning Theory (ALT), SLT analyzes the statistical properties of the data to construct computational models. In other words, SLT considers the properties of the data and the functions, while ALT considers the properties of the algorithms which *generate* functions and operate on the data.

In this work, first statistical properties of the data and the algorithms have been analyzed. Then, upon these analyses, new methods are proposed for Data Fusion of Statistical Learning methods, specifically for Decision Fusion of Supervised, Unsupervised and Semi-supervised Learning.

Mathematically speaking, let X and Y be the probability spaces of all possible inputs and outputs to an algorithm in the domain and range of functions $f \in \mathfrak{F}$, respectively. $\mathfrak{F}$ is called a function or an hypothesis space, and X and Y are called data spaces. Assume that a set of samples $S = \{s_i : s_i = (x_i, y_i)\}_{i=1}^{N}$, whose elements are distributed by an unknown but a joint distribution $p(x, y)$ in a probability space $X \times Y$, is given. Denote the *difference* between the target y_i and a function on an observation datum x_i as $\epsilon(f(x_i), y_i)$, which is called an error function of f on s_i.

There are various constraints imposed on the distribution of the samples. For instance, a widely employed assumption is that the samples are independent and identically distributed

(i.i.d.) with joint probability density function $p(x, y)$. The joint distribution, and cumulative distributions of the observations and targets are usually assumed to be stationary, i.e. they do not change in time. If the distributions change in time, then new learning problems, such as Data-shift and Domain-Shift, are defined and analyzed by Data Adaptation or Domain Adaptation methods [20, 21, 22, 23, 24, 25]. For the sake of simplicity, stationary distributions are assumed in this work.

Vapnik defines the problem of statistical learning or pattern recognition as the computation of the function $f : X \rightarrow Y$ which captures the relationships between observations and targets by minimizing the expected error $E\{\epsilon(f(x_i), y_i)\}$ [26, 27, 5]. The motivation of the error minimization is to compute an hypothesis function which *generalizes* on a new given sample *well* using the dataset S.

There are various other approaches studied in SLT that are also related to ALT, such as Valiant's Probably Approximately Correct (PAC) learning [28] which analyzes the *learnability* of the algorithms measuring capacity or size of the hypothesis spaces generated by the algorithms. Therefore, ALT and SLT meets for the analysis of the functions employed on the data in some learning models such as PAC.

The statistical learning problems that have been studied in this work are grouped according to the availability of information on the targets. In addition, the algorithms that are proposed to solve these problems have been analyzed considering the statistical properties of the data spaces. In the literature, three statistical learning methods, namely Supervised, Unsupervised and Semi-supervised learning, have been studied to solve the problems considered under that categorization.

In the next subsection, an overview of the challenges of SLT which have been addressed in this work is given with these three learning methods.

2.1.1 Challenges of Statistical Learning Methods

Among various *exciting* and *delightful* problems of SLT, two challenges have been studied in this work. The definitions and the methods which aim to solve these problems are categorized according to the assumptions and approaches to analyze the probability spaces and the functions, defined on these spaces.

The first problem is to find an *optimal* representation of sample measurements in data spaces for a given algorithm. This problem has been studied in the literature through the feature extraction methods in machine learning and pattern classification [4]. There are major problems of feature extraction; *i*) an optimal representation of the samples may not exist in a single feature space, and *ii*) a single algorithm may not be able to process the features in a single feature space even if such a representation exists.

In this work, we solve the above mentioned feature extraction problems simultaneously unifying them under a Decision Fusion framework. Instead of searching the *best* feature set for the representation of the samples, multiple learning algorithms have been employed on different feature spaces. Then, the outputs or *decisions* of the algorithms are fused to obtain a final decision.

The second problem is to define an *optimal* distance function which is to be computed between

the feature vectors. This problem is studied under the heading of distance learning in the literature [29, 30]. There are two common approaches for distance learning:

1. In the first approach, distance functions are *designed* according to the specific problem of interest and their parameters are learned from the datasets.
2. In the second approach, either
   (a) feature spaces are mapped to more *informative* feature spaces, or
   (b) the features in feature spaces are weighted to obtain more *informative* features.

   Then, the mapping parameters or the weights are learned from the datasets by preserving the *structure* of the distance function. Note that a and b may be equivalent in some special cases, e.g. when the mappings are linear and parameterized by the weights.

The first approach, i.e. distance function design, has been used to solve unsupervised clustering or segmentation problem. In addition, several weighted decision fusion algorithms have been proposed by employing the second approach, i.e. feature space mapping and fusion, for Unsupervised, Supervised and Semi-supervised Learning.

In addition to the employment of these two approaches, a new third approach is proposed in this work for distance learning. In the proposed approach, the distance learning problem is defined as the problem of designing classifiers and decision fusion methods. For this purpose, first the individual learning algorithms in a hierarchical decision fusion algorithm maps the input feature spaces into decision space as suggested in (a). Then, decisions in decision spaces are aggregated with weights to construct fusion spaces. This operation can be used as a non-linear mapping from decision spaces to fusion spaces (a), and a linear weighted feature mapping in fusion spaces (b). In this case, the proposed approach combines feature space mappings in (a) and feature space weights in (b). A detailed explanation of the proposed solutions are given in the following subsection with discussions. Mathematical representation of the solutions and algorithms are given in Part I and II.

2.1.2 Supervised Learning Methods

In Supervised Learning, information on targets is obtained from a set of target variables belonging to a training dataset S. If the target variables take values from real number space $\mathbb{R}$, then the learning problem is called regression, or function learning [31, 32]. If the target variables take values from integer number space $\mathbb{Z}$, then the learning problem is called classification or categorization [4].

There are two common learning models in Supervised Learning, namely Generative and Discriminative Learning. In Generative Learning, the interest is on the computation of $p(x, y)$ such that the samples can be *generated* when the joint distribution is estimated. Vapnik criticizes this model by stating that "one should solve the problem directly and never solve a more general problem as an intermediate step" [26, 27]. The second model, Discriminative Learning, follows the suggestion of Vapnik by directly estimating $p(y|x)$. Although there are several heuristics to prefer one of the models to the other, the choice depends on the confidence in the correctness of computing $p(x, y)$ or $p(y|x)$ using the training dataset [33]. Another approach is constructing Hybrid Generative-Discriminative models [34]. In this approach, first

a probabilistic relation or model between the latent or hidden variables of observations and targets is learned using Generative Learning on the training dataset. Then, the models are combined to predict the targets using Discriminative Learning.

In this work, one of the most stable, well-known, *powerful* and the simplest Discriminative Supervised Learning models, k Nearest Neighbors (k-NN) [12], has been analyzed and used in an ensemble of classifiers where the outputs are aggregated for Decision Fusion. In k-NN, for a given sample point s, first its k nearest neighbors are found. Then, the target values or the decisions of the nearest neighbors have been fused using a fusion rule, such as majority voting, to estimate the target value or the decision of s.

Despite its simplicity, there are various open problems about the behavior of k-NN which have been studied for a long decade and not been solved yet, such as the computation of a distance function which minimizes the N-sample or large-sample classification error, or their difference.

In this thesis, distance learning for k-NN has been analyzed by defining the distance learning problem as a Decision Fusion problem. For this purpose, a distance learning approach for k-NN is implemented in a Hierarchical Decision Fusion algorithm consisting of various modified k-NN algorithms employed on different feature spaces at each different layer of the hierarchy. Then, decisions of k-NN algorithms employed in the base-layer have been fused to provide a feature space to another k-NN classifier. which computes the final decision using at the meta-layer. It is shown that the suggested method minimizes the *difference* between the N-sample and large-sample classification error of 1-NN algorithm in the next chapter.

In one sense, k-NN is itself a *micro-level* Decision Fusion algorithm. Because, the label of a sample is estimated by fusing the decisions of its neighbor samples, which are defined by the labels of neighbor samples. Therefore, the employed Hierarchical Decision Fusion algorithm can be considered as a *Multi-scale* Decision Fusion algorithm.

The second challenge that has been studied in this thesis is to minimize the large-sample classification error of k-NN, which converges to Bayes Error, using a decision fusion approach. For this purpose, two conditions stated by Cover and Hart [13] have been analyzed; i) the class posterior probabilities should be estimated with maximum certainty, ii) the estimation should be made for each sample in the dataset.

In order to satisfy the first condition (i), the contribution of each sample to the classification performances of base-layer and meta-layer classifiers are measured using a new distance function called Decision Margin. It is shown that if a sample is correctly classified with maximum certainty by a base-layer classifier, then its decision margin value computed at the output *decision space* of the base-layer classifier is minimum. Moreover, if a sample is correctly classified by each of the base-layer classifier with minimum decision margin, then meta-layer classifier estimates the posterior probability, i.e. class label, of the sample with maximum certainty in a *fusion space* which is the Cartesian product of decision spaces.

In order to minimize decision margins of the samples in fusion space, a weighted decision fusion algorithm, which assigns weights to decision spaces according to classification performances of base-layer classifiers and decision margins of the samples, is proposed. Decision weights are computed using a convex optimization algorithm which is given in Section 3.5.2.

In order to satisfy the second condition (ii), two spurious sample selection and elimination algorithms are proposed. Unlike the sample selection method of Wilson [35] for k-NN, elimina-

tion of spurious samples have been employed in various levels of the hierarchy in the proposed algorithms which are given in Section 3.5.3.

2.1.3 Unsupervised Learning Models

Unsupervised Learning differs from Supervised Learning in the accessibility to the information on the targets such that only the observations are considered in a training dataset $S_u = \{s_i : s_i = (x_i)\}_{i=1}^{N}$, and target variables are not available to the algorithms. Therefore, determination of the variables of the error functions depends on the problem definition, which is usually ill-posed, such as clustering [4].

Although the definition of the clustering problem is as simple as *grouping a set of samples* in clusters in a feature space, it is a challenging problem. Because, there is no well defined definition of an *optimal* or a *target group*, *cluster* and feature *representation* of samples which is universally consistent. Some widely studied specializations of the problem are,

- Finding clusters of sample groups in data spaces,
- Grouping the samples in clusters, where samples within each cluster are *close* to each other or the cluster center, and samples of different clusters are *far* from each other or these cluster centers.

The first problem, finding the clusters of sample groups, is still an ill-posed problem, especially when the number of different clusters (i.e. cluster names or labels C) and the target clusterings are not available in the training dataset. The estimation of the number of different clusters has been analyzed in Chapter 5.

The second problem can be considered as a vector quantization problem, such that the vector representation of the samples in data spaces, called *features*, are mapped to one of C number of different clusters, i.e. quantized by the index of the clusters. Note that this is strongly related to Coding Theory by representing samples using *codes* generated by the clustering algorithms. Therefore, it is not surprising that one of the most successful clustering algorithms C-means (or k-means) has been studied first for analyzing Pulse-Code Modulation problem [36].

C-means is a center based clustering algorithm. A brief description of the algorithm is the following;

- **Initialize**: Randomly choose C cluster centers in the data space in which samples are represented by features.
- **do**
  - Assign each sample s to a cluster $c = 1, 2, \ldots, C$ if a *distance* between the feature of s is closer to the center of c than the centers of the other clusters.
  - Update the center of the clusters as the mean of the features of the samples assigned to it.

  while there is no further update in the assignments and cluster centers.

Despite its simplicity, C-means is a *hard* problem in terms of computation. Clustering using C-means in an *arbitrary* dimensional Euclidean feature space for a *fixed* $C = 2$ number of clusters, or in a *fixed* dimensional (e.g. two dimensional) feature space with *arbitrary* number of clusters is an NP-Hard problem [37, 38, 39]. If the dimension of feature spaces and the number of clusters are fixed, then the problem can be solved *exactly* in polynomial time [40].

In addition to the computational *hardness* of clustering, designing feature spaces and distance functions are two other challenges. Instead of designing the *best* feature space on which a clustering algorithm is employed with the *minimum error*, fusion of the outputs or clusterings of an ensemble of clustering algorithms [41] employed on individual feature spaces are studied in this work. For this purpose, a special error criterion, which achieves a consensus among the algorithm decisions, is suggested. In the literature, combination or the fusion of the clustering algorithms to achieve the consensus among them has been studied as Consensus Clustering [42].

In the proposed framework, the distance learning problem is handled by two approaches. First, structure and parameters of distance functions employed in the fusion of outputs of clustering algorithms are *learned* from the training datasets. Second, the problem is defined as a weighted clustering fusion problem and the weights are computed using the training datasets.

In this work, unsupervised clustering algorithms are studied under an image segmentation framework. The employment of clustering algorithms to image segmentation problem is straightforward with two modifications. First, sample features are color vectors representing image pixels with a specific coordinate. Therefore, the clusters are called segments and the clustering results are called segmentations. Second, spatial relationships among the pixels *should be* considered in the design of feature spaces.

The second modification is required in order to obtain segmentations which are *semantically* and *cognitively meaningful* to humans. However, the clustering algorithms do not satisfy the necessary and sufficient conditions to achieve *meaningful* segmentations. For instance, a consensus segmentation algorithm may converge to an *optimal* solution, which provides a segmentation that resides in a decision space at equal distance to each segmentation obtained from each individual segmentation algorithm. However, the *consensus solution* may not be *meaningful* to humans. Therefore, side information about a target *meaningful* segmentation should be incorporated both in the error function and distance function of the algorithm. Considering the definitions of clustering problem given above, *must-link* and *cannot-link* constraints can be used as side information about the co-existence of the two pixels with the same labels in the same segment and to avoid the co-existence of the two pixels with different labels in the same segment, respectively. Incorporation of side information into Unsupervised Learning algorithms have been studied by Semi-supervised Learning Methods.

2.1.4 Semi-supervised Learning Methods

Semi-supervised Learning methods can be analyzed from two different perspectives. From the perspective of Supervised Learning, the partial information on the unlabeled test samples are used during the computation of the learning models in Semi-supervised Learning [6]. On the other hand, side information obtained from labeled samples, such as the co-occurrence of the samples belonging to the same class in the same cluster, is incorporated to problem definition from the perspective of Unsupervised Learning. Therefore, Semi-supervised Learning methods

do not reside neither in the intersection nor in the union of Supervised and Unsupervised Learning methods; rather, they should be considered as a separate paradigm.

In Semi-supervised Learning, a restricted set of labeled training samples S and a set of unlabeled training samples S_u are available to the algorithms. Indeed, training with unlabeled samples together with labeled samples may not help improving the performance [43]. Therefore, additional assumptions should be made on data and learning models [6].

In this work, semi-supervision is employed to unsupervised image segmentation problems. Therefore, three assumptions about the side information have been made in this work;

1. There exist *discrete* segments in the datasets, i.e. the pixels can be split into groups by analyzing data or feature spaces.
2. Pixels in the same segment are more likely to share a label. In other words, must-link constraints do not violate the learning model and can be employed on the datasets.
3. Pixels in different segments are more likely to have different labels. In other words, cannot-link constraints do not violate the learning model and can be employed on the datasets.

The major goal of this thesis is to develop methods which combine individual Supervised, Unsupervised and/or Semi-supervised learning algorithms to boost the overall performance of the fusion architectures. In the next section, we present an overview of the available data fusion approaches emphasizing power and weakness of them. This discussion prepares a background for our motivation for the suggested Supervised, Unsupervised and Semi-supervised Data Fusion methods.

2.2 An Overview of Data Fusion Approaches

With the proliferation of different sensors and information sources providing information about certain phenomenon, data/information fusion has become extremely important. It is well-known that the available data fusion strategies independently fuse the data, features or decisions at different levels, optimizing local error functions. One of the major drawbacks of the current approaches is the lack of techniques to optimize the parameters of data, features and/or decisions across the levels, which are needed for global optimization of the fusion parameters. In other words, once a strategy performs fusion at data and/or feature and/or decision levels according to the parameter optimization of a certain level, then, the strategy cannot be updated or revisited for further improvements of the overall or local performances at the upper levels.

Multiple sensors, features, and/or classifiers are used for data fusion in data, feature and decision levels, using hierarchical algorithms [44]. The goal of these methods is to extract information from different data sources and feature spaces, which cannot be obtained from single data source or feature space, using multiple classifiers or decision systems [45].

In the literature, Data Fusion is considered to be accomplished at 3 basic Levels:

1. Data Level,
2. Feature Level and
3. Decision Level.

In the following subsections, fusion of data, features and decisions at the above mentioned levels will be described and the major problems associated with fusion approaches will be discussed.

2.2.1 Data Level Fusion

The main concern of Data Level Fusion is to process the raw data obtained from multiple sensors in order to extract compact and valuable information. For this purpose, the data obtained from the sensors are processed by various classical techniques, such as estimation, detection and registration [9, 10, 11]. The techniques, implemented at this level, highly depend on the application domain. For example, in the remote sensing applications, pixel based fusion methods are employed [46], on the other hand, detection and estimation methods are common in target detection problems [47].

Rottensteiner *et al.* [48] aggregate data obtained from laser scanner and multi-spectral image sensors for building detection. Turlapaty *et al.* use multi-angular optical data to estimate height information and extract buildings [49]. Fabre, Briottet and Appriou [50] employ pixel fusion using Dempster-Shafer theory. Al-Osaimi et al. [51] use pixel level fusion of 3-D shape and texture for face recognition. Chibelushi et al. [52] give an overview of data fusion approaches for speech recognition.

There are two major problems associated with Data Level Fusion. The first problem is the high complexity of the analysis and the evaluation of physical models of the sensors which provide the data [53, 54]. In practice, the estimation and the detection of the physical model of each sensor and type of data may not be available. In addition, these approaches suffer from the construction of appropriate models for the targets and the mismatches between the real and the modeled data. The second problem is the high computational complexity for the estimation of the models of each type of data and sensor. A detailed analysis of the above mentioned problems is provided in [47, 55, 56].

2.2.2 Feature Level Fusion

The second level of data fusion is Feature Level Fusion. At this level, multiple features are extracted from a single sensor data or from the fused data coming from data level fusion. The extracted features may be preprocessed by normalization and/or via dimensionality reduction techniques [57, 58, 59, 60]. Finally, the features are fused to create feature groups [9].

Li *et al.* [61] combine spectral and texture features for the detection of collapsed buildings which are damaged by an earthquake. Hansch and Hellwich [62] use a SLT algorithm called

Random Forests to combine polarimetry, intensity and texture features extracted from Polarimetric Synthetic Aperture Radar (SAR) images for building detection. Fernando et al. [63] proposed a discriminative image classication method using feature level fusion. Wender and Dietmayer [64] fused the features extracted from laser scanner and video data for moving object classification. Sharma and Davis [65] combine contour features extracted from thermal and visible sensor data for image segmentation.

The current data fusion approaches implement the Data and the Feature Level Fusion in a sequential manner. The data obtained from Data Level Fusion directly affects the structure and the quality of Feature Level Fusion. Therefore, the information represented at the output of the Feature Level Fusion depends on the Data Level Fusion strategies. The problems and errors inherent to the Data Level and Feature Level Fusion propagate to the upper levels affecting the performance of the overall architecture.

2.2.3 Decision Level Fusion

The upper level of a Data Fusion architecture is Decision Level Fusion, which receives the feature groups from the Feature Level and outputs the final decision. There are various techniques and architectures in order to perform the decision inferences [8].

A major problem in Decision Level Fusion is the requirement of a priori information on the features. In most of the applications, the a priori information cannot be estimated easily or may not be available. One of the reasons for this problem is that the decision process is based only on the features obtained at the output of feature level fusion, ignoring the information on the data and feature structures at the lower levels.

The information on the features may include crisp or fuzzy hypothesis sets on the class conditional distributions of the samples [9]. The hypothesis vectors construct the output feature space of a decision fusion algorithm. This methodology is commonly used for ensemble learning which aims to provide a priori information on the samples to the meta-layer classifiers at the Decision Level [66]. Various probabilistic, possibilistic (i.e., fuzzy), and crisp decision fusion methods are analyzed by Huang and Zhang [67], and by Milisavljevic and Bloch for object detection and classification [68].

Decision Fusion methods have been integrated with and related to the statistical learning methods. Therefore, a detailed discussion on the relationship is given in the next section.

2.3 Statistical Learning and Data Fusion

Data fusion problems have been studied as the challenges of statistical learning algorithms, such as designing the classification rules that perform better than other classifiers, or a feature set that represents the samples better than the other features.

One approach to solve this problem is to combine an ensemble of classifiers to boost the performance of the individual classifiers in the ensemble. There are many algorithms available in the literature for ensemble learning which can be studied under the heading of Decision Level Fusion [66, 69, 70, 8, 71, 72, 73, 74].

Another approach is to collect data by using more than one sensor and extract multiple features to represent the samples. This approach can be studied under the heading of Feature Level Fusion in the literature [57, 58, 59, 60, 9, 61, 62, 63, 64, 65].

Statistical learning algorithms have been widely used by several researchers for object detection and tracking at different fusion levels [75, 76, 77, 78, 71, 79, 80, 69, 81]. Fu, Cao, Guo, and Huang [75] perform feature level fusion for face detection by transforming the feature spaces into a common vector space and decrease the dimension of the space by PCA while considering the maximum mutual information correlation of the feature spaces. However, this fusion method suffers from 3 problems. First the method does not consider the isometry of the space transformations while reducing the vector spaces with different levels of sparsity, which causes the unbalanced vector topology and dominated component construction for the projection space. Second the method evaluates different spaces with multiple modalities by the assuming that the spaces have the same physical structure with the same topological behavior. However, this assumption is not acceptable for multimodal features, especially for audio data in frequency domain and video data in space domain. Therefore, the probabilistic correlation of the feature spaces may generate false alarms. Finally, they perform learning phase after the transformation of the spaces by ignoring feature level training in order to obtain inference representing the statistical behavior of the feature spaces from the data.

Dai, Yang, Wu and Katsaggelos [76] perform component based detection by applying logical set operations such as AND and OR on the feature subspaces for fusion operations at feature level. However, this method suffers from the audio-visual sensor fusion because of the subspace construction problem for the audio data, where audio subspaces should be well matched with video subspaces.

Polikar [77] and Avidan [78] apply boosting as an ensemble method for data fusion at decision level. However, firstly, the methods suffer from the problems of the Adaboost such as over-fitting [82], noise sensitivity and computational complexity [71]. Secondly, applying voting (majority or weighted) as a fusion mechanism for boosting may not be suitable for audio-visual data fusion, since the voting requires the correct classification of the training data by at least greater than half of the number of the weak classifiers without regarding the topological structure of the data [71].

Fauvel *et al.* [69] propose a decision fusion approach in which the features are extracted by morphological operators. Then, the features are classified using neural networks and a fuzzy classifier. Finally, posterior probabilities obtained from the output of the neural networks and the class membership degrees obtained from the fuzzy classifier are fused. Sun *et al.* [81] propose an interpretation model for the classification of targets in urban areas. They extract the segments using pyramid-cut, and then compute color, texture, shape and location features for each segment. Finally, they classify the segments by a Joint Boosting algorithm. Waske and Benediktsson [83] aggregate the decisions of individual SVMs which process multitemporal synthetic aperture radar data and optical images independently using a majority voting method for multi-sensor data classification.

Another statistical learning algorithm employed for Decision Fusion is Stacked Generalization (SG), which is proposed by Wolpert [84] and used by many others [85, 86, 87, 88, 74, 89, 90, 91, 92, 93]. The basic idea is to ensemble several classifiers in a variety of ways so that the performance of the SG is higher than that of the individual classifiers which take place under the ensemble.

There are various approaches to implement the Stacked Generalization architecture. Ueda [86] employs feature vector concatenation operation at the output of the base-layer feature spaces of SG and call these operations as the linear decision combination methods. Then, he compares linear combination and voting methods in an SG architecture, experimentally, where Neural Networks are implemented as the base-layer classifiers. Following the same formulation, Sen and Erdogan [87] analyzes various weighted and sparse linear combination methods by combining the decisions of heterogeneous base-layer classifiers such as decision tree and k-NN method. Rooney et al. [88] employs homogeneous and heterogeneous classifier ensembles for stacked regression using linear combination rules. Zenko et al. [74] compares the classification performances of SG algorithms, which employ linear combination rules with the other combination methods (e.g. voting) and ensemble learning algorithms (e.g. Adaboost).

Sigletos et al. [89] compare the classification performances of several Stacked Generalization algorithms which combine nominal (i.e., crisp decision values such as class labels) or probabilistic decisions (i.e., estimations of probability distributions). In most of the experiments given in the above mentioned studies, linear combination or decision aggregation method provides comparable or better performances than the other combination methods.

However, the performance evaluation of the stacked generalization methods reported in the literature are not consistent with each other. This fact is demonstrated by Dzeroski and Zenko in [90] where they employ heterogeneous base-layer classifiers, in their stacked generalization architecture. They report that their results contradict with the observations of the studies in the literature on SG. The contradictory results can be attributed to many nonlinear relations among the parameters of the SG, such as the input feature spaces, structure of the heterogeneous classifiers and their output feature spaces.

For instance, popular classifiers, such as, k-NN, Neural Networks and Naïve Bayes, can be used as the base-layer classifiers in SG to obtain nominal decisions. However, there are crucial differences among these classifiers in terms of processing the feature vectors. First, k-NN and Neural Networks are non-parametric classifiers, whereas the Naïve Bayes is a parametric one. Second, k-NN is a local classifier which employs the neighborhood information of the features, whereas Neural Networks compute a global linear decision function and Naïve Bayes computes the overall statistical properties of the datasets. Therefore, tracing the feature mappings from base-layer input feature spaces to meta-layer output decision spaces in SG becomes an intractable and uncontrollable problem. Additionally, the outputs of the heterogeneous classifiers give different type of information about the decisions of the classifiers, such as crisp, fuzzy or probabilistic class labeling.

Although gathering the classifiers under the SG algorithm significantly boosts the performance in some application domains, it is observed that the performance of the overall system may get worse than that of the individual classifiers in some other cases. In fact, the performance may even get worse and worse, as the number of the classes and the dimension of the feature space increases. Wolpert defines the problem of describing the relation between the performance and various parameters of the algorithm as a *black art* problem [84, 85].

2.3.1 Decision Fusion for Supervised Learning

In this study, most of the above mentioned intractable problems of supervised learning have been analyzed by employing a hierarchical learning algorithm called Fuzzy Stacked General-

ization (FSG) [94, 95, 70] which consists of the same type of classifiers in SG. The proposed technique fuses the independent decisions of the fuzzy base-layer classifiers by aggregating the membership values of each sample for each class under the same vector space, called the decision space. A meta-layer fuzzy classifier is, then, trained to learn the degree of the correctness of the base-layer classifiers.

This architecture allows the fusion of the output decision spaces of the base-layer classifiers, which contain consistent information. Furthermore, linear combination or feature space aggregation method is modeled as a feature space mapping from the base-layer output feature space to the meta-layer input feature space. In the proposed model, classification rules are considered as the feature mappings from classifier input feature spaces to output decision spaces. In order to control these mappings for tracing the transformations of the feature vectors of the samples through the layers of the architecture, homogeneous classifiers are used in both the base-layer and the meta-layer.

The employment of fuzzy decisions in the ensemble learning algorithms is analyzed in [91, 92, 93]. Tan et al. [91] uses fuzzy k-NN algorithms as base-layer classifiers, and employs a linearly weighted voting method to combine the fuzzy decisions for Face Recognition. Cho and Kim [92] combine the decisions of Neural Networks which are implemented in the base-layer classifiers using a fuzzy combination rule called fuzzy integral. Kuncheva [93] experimentally compares various fuzzy and crisp combination methods, including fuzzy integral and voting, to boost the classifier performances in Adaboost. In their experimental results, the algorithms which implement fuzzy rules outperform the algorithms which implement crisp rules. However, the effect of the employment of fuzzy rules to the classification performances of SG remains an open problem.

In the proposed architecture, the fuzzy k-NN method is employed and the behavior of fuzzy decision rules are investigated at the base-layer and meta-layer classifiers. Moreover, employment of fuzzy k-NN classifiers enables us to obtain information about the uncertainty of the classifier decisions, and the belongingness of the samples to classes [96, 97].

2.3.2 Decision Fusion for Unsupervised and Semi-supervised Learning

Unsupervised and Semi-supervised Learning methods employed for Decision Fusion can be studied in two groups. The works belonging to the first group employ statistical signal processing methods such as Dempster-Shafer Theory [98, 99, 100], Causal Autoregressive Random Fields [101] or Markov Random Fields [102].

The works in the second group aggregate the decisions of several base-layer unsupervised learning algorithms as in supervised decision fusion methods. In Machine Learning community, this approach is referred as Ensemble or Consensus Clustering [41, 103, 104] as mentioned in the previous sections. Consensus clustering problem has been studied as the median partition problem, in which a partition is computed so that the *average distance* between this proposed partition and the elements of a set of given partitions obtained from base-layer clustering algorithms is minimized. However, finding a globally optimal solution using a symmetric difference distance is an NP-complete problem [18].

In order to solve the problem in polynomial time, two greedy search algorithms, Best of K (BOK) and Best One Element Move (BOEM), are proposed [18]. In BOK, the distance is

computed for each partition which is sequentially selected as the consensus partition from a set of K partitions. Then, a partition with minimum distance to all the other partitions is selected as the *best* of K partitions. In BOEM, first an initial partition is selected using an algorithm such as BOK. Then, only one-element of the current partition (i.e. one element of *best partition*) is moved at each iteration of the algorithm as long as the distance of the partition to the other partitions decreases.

Yang et al. used semi-supervision for the fusion of the decisions of unsupervised algorithms with supervised algorithms [73]. Franek et al. [105] and Vega-Pons et al. [106] employed various state of the art Ensemble Clustering algorithms for image segmentation and introduced a detailed experimental analysis on Berkeley Image Segmentation Dataset [17]. They report that, among various clustering or segmentation fusion algorithms, two *less complex* algorithms, Best One Element Move (BOEM) and Best of K (BOK) algorithms have provided competitive results.

Motivated by the performances of consensus clustering algorithms in image segmentation problems, a new Segmentation Fusion (SF) algorithm is introduced in this work. In SF, first the consensus clustering problem is reformulated as the segmentation fusion problem. Then, the problem is solved using a filtered stochastic optimization method called Filtered Stochastic BOEM [107].

In this study, two challenges of SF, which are learning the distance function from the data, and the estimation of the algorithm parameter and the number of different segment labels C, have been studied. For this purpose, two distance learning functions and two parameter estimation methods are proposed. In addition, weighted distance learning problem is defined as the weighted decision fusion problem and a method is suggested to solve the problem by learning weights from the data.

Incorporating prior information about the data statistics and side information obtained from expert knowledge is another challenge of consensus segmentation and clustering algorithms. This problem has been studied for subspace clustering [108], and generalized cluster aggregation [109] problems. Together with these challenges, sparse weighted decision fusion with semi-supervision and distance learning have been considered in a single constrained optimization problem in this work. The proposed problem has been analyzed and solved using a Semi-supervised Segmentation Fusion (SSSF) algorithm.

Part I

Decision Fusion for Supervised Learning

CHAPTER 3

CLASSIFICATION ERROR MINIMIZATION IN DECISION FUSION

3.1 Introduction

In this chapter, the classification error minimization problem has been analyzed from the Decision Fusion perspective. A well-known Decision Fusion approach for minimizing the classification or the generalization error is, first designing a set of *weak classifiers* by minimizing the training error, and then combining the classification models or decisions of the weak classifiers. This approach has been studied under the ensemble learning algorithms for a long decade [93] to solve various practical problems, such as target detection, localization [47, 8, 9, 10], object recognition and classification [10, 46, 55], and discussed in the previous chapter.

In this chapter, the classification error of a k-NN classifier is analyzed in two parts, namely i) N-sample error which is the error of a classifier employed on a training dataset of N samples and ii) large-sample error which is the error of a classifier employed on a training dataset of large number of samples such that $N \rightarrow \infty$. First, a distance learning approach which minimizes the error difference between N-sample and large-sample error is analyzed. Then, the distance learning approach is employed in a hierarchical Decision Fusion algorithm, called Fuzzy Stacked Generalization (FSG) [94, 95, 70]. Finally, a weighted decision fusion and two sample selection algorithms are suggested to minimize the large-sample error.

In the next section, N-sample and large-sample classification errors of k-NN are defined. A distance learning approach proposed by Short and Fukunaga [14] which minimizes the error difference between N-sample and large-sample errors for a k-NN algorithm is given in Section 3.3. The employment of the distance learning approach in FSG is explained in Section 3.4. A weighted decision fusion algorithm and two sample selection algorithms for FSG, which minimize the large-sample error, are given in Section 3.5. Computational complexities of the suggested algorithms are analyzed in Section 3.6. Experimental analyses of the proposed algorithms are given in Section 3.7. Section 3.8 summarizes the analyses, the proposed algorithms and the results, given in the chapter.

3.2 N-sample and Large-sample Classification Errors of k-NN

Suppose that a training dataset $S = \{(s_i, y_i)\}_{i=1}^{N}$ of N samples, where $y_i \in \{\omega_c\}_{c=1}^{C}$ is the label of a sample s_i, is given. A sample s_i is represented in a feature space F_j by a feature vector $\bar{x}_{ij} \in \mathbb{R}^{D_j}$.

Let $\{P(\bar{x}_j|\omega_c)\}_{c=1}^{C}$ be a set of probability densities at a feature vector $\bar{x}_j$ of a sample s, such that $\bar{x}_j$ is observed by a given class label ω_c according to density $P(\bar{x}_j|\omega_c)$. Therefore, $P(\bar{x}_j|\omega_c)$ is called the likelihood of observing $\bar{x}_j$ for a given ω_c. A set of functions $\{P(\omega_c)\}_{c=1}^{C}$ is called the set of prior probabilities of class labels such that $\sum_{c=1}^{C} P(\omega_c) = 1$ and $P(\omega_c) \geq 0$, $\forall c = 1, 2, \ldots, C$. Then, the posterior probability of assigning the sample s to a class ω_c in F_j is computed using the Bayes Theorem [4] as

$$P(\omega_c|\bar{x}_j) = \frac{P(\bar{x}_j|\omega_c)P(\omega_c)}{\sum_{c=1}^{C} P(\bar{x}_j|\omega_c)P(\omega_c)}.$$

Note that $\bar{x}_j$ is a random vector which maps the prior probability vector

$$[P(\omega_1), \ldots, P(\omega_c), \ldots, P(\omega_C)]$$

into the posterior probability vector

$$[P(\omega_1|\bar{x}_j), \ldots, P(\omega_c|\bar{x}_j), \ldots, P(\omega_C|\bar{x}_j)].$$

Bayes classification rule estimates the class label $\hat{y}$ of s as [4]

$$\hat{y} = \arg\max_{\omega_c} \{P(\omega_c|\bar{x}_j)\}_{c=1}^{C}.$$

If a loss $\mathcal{L}(\omega_c, s)$ occurs when a sample s is assigned to $\hat{y} = \omega_c$, then the classification error of the Bayes classifier employed on F_j is defined as [13]

$$err(s) = \min_{c} \sum_{c=1}^{C} \mathcal{L}(\omega_c, s)P(\omega_c|\bar{x}_j)$$

and the expected error is defined as [13]

$$e* = E\{err(s)\},$$

where the expectation is taken over the density $p(\bar{x}_j)$ of the feature vectors in F_j.

In this work, we focus on the minimization of the classification error of a well-known classifier which is k Nearest Neighbors (k-NN) [4]. Given a new test sample (s', y') with $\bar{x}'_j \in F_j$, let $\aleph_k(\bar{x}'_j) = \{\bar{x}_{l(1)j}, \ldots, \bar{x}_{l(k)j}\}$ be a set of k nearest neighbors of $\bar{x}'_j$ such that

$$d(\bar{x}'_j, \bar{x}_{l(1)j}) \leq d(\bar{x}'_j, \bar{x}_{l(2)j}) \leq \ldots \leq d(\bar{x}'_j, \bar{x}_{l(k)j}).$$

The nearest neighbor rule (e.g. $k = 1$) simply estimates $\hat{y'}$, which is the label of $\bar{x}'_j$, as the label $y_{l(1)}$ of the nearest neighbor $\bar{x}_{l(1)j}$ of $\bar{x}'_j$. In the k nearest neighbor rule (e.g. k-NN), $\hat{y'}$ is estimated as

$$\hat{y'} = \arg\max_{\omega_c} \mathcal{N}(\aleph_k(\bar{x}'_j), \omega_c),$$

where $\mathcal{N}(\aleph_k(\bar{x}'_j), \omega_c)$ is the number of samples which belong to ω_c in $\aleph_k(\bar{x}'_j)$.

Then, the probability of error $\epsilon(\bar{x}_{i,j}, \bar{x}'_j) = P_N(error|\bar{x}_{i,j}, \bar{x}'_j)$ of the nearest neighbor rule is computed using N number of samples as

$$\epsilon(\bar{x}_{i,j}, \bar{x}'_j) = 1 - \sum_{c=1}^{C} \mu_c(\bar{x}_{i,j})\mu_c(\bar{x}'_j), \tag{3.1}$$

where $\mu_c(\bar{x}_{i,j}) = P(\omega_c|\bar{x}_{i,j})$ and $\mu_c(\bar{x}'_j) = P(\omega_c|\bar{x}'_j)$ represent posterior probabilities [4].

In the asymptotic of large number of training samples, if $\mu_c(\bar{x}'_j)$ is not singular, i.e. continuous at $\bar{x}'_j$, then large-sample error $\epsilon(\bar{x}'_j) = \lim_{N\to\infty} P_N(error|\bar{x}'_j)$ is computed as

$$\epsilon(\bar{x}'_j) = 1 - \sum_{c=1}^{C} \mu_c^2(\bar{x}'_j). \tag{3.2}$$

It is well known that there is an elegant relationship between the classification errors of Bayes classifier and k-NN as follows [13]:

$$e* \leq \epsilon(\bar{x}'_j) \leq \epsilon(\bar{x}_{i,j}, \bar{x}'_j) \leq 2e*.$$

Then, the difference between the N-sample error (3.1) and the large-sample error (3.2) is computed as

$$\epsilon(\bar{x}_{i,j}, \bar{x}'_j) - \epsilon(\bar{x}'_j) = \sum_{c=1}^{C} (\mu_c(\bar{x}'_j))(\mu_c(\bar{x}_{i,j}) - \mu_c(\bar{x}'_j)). \tag{3.3}$$

In this chapter, we have two goals. The first goal is to minimize the difference between $\epsilon(\bar{x}_{i,j}, \bar{x}'_j)$ and $\epsilon(\bar{x}'_j)$ (3.3) by employing a distance learning approach suggested by Short and Fukunaga [14] using Fuzzy Stacked Generalization. The distance learning approach of Short and Fukunaga and its employment in the Fuzzy Stacked Generalization algorithm is given in Section 3.3. The second goal is to minimize the difference between $\epsilon(\bar{x}'_j)$ and $e*$ using a new distance function called Decision Margin in a new algorithm called weighted decision fusion. This algorithm reveals two sample selection methods for the FSG, which are given in Section 3.5.

3.3 Minimization of N-sample and Large-sample Classification Error Difference using Distance Learning in the FSG

Let us start by defining

$$e_c(\bar{x}_{i,j}, \bar{x}'_j) = \left(\mu_c(\bar{x}_{i,j}) - \mu_c(\bar{x}'_j)\right)^2$$

and an error function

$$e(\bar{x}_{i,j}, \bar{x}'_j) = \sum_{c=1}^{C} \epsilon_c(\bar{x}_{i,j}, \bar{x}'_j)$$

for a fixed test sample $\bar{x}'_j$. Then, the minimization of the expected value of the error difference in (3.3), $E_N\{\left(\epsilon(\bar{x}_{i,j}, \bar{x}'_j) - \epsilon(\bar{x}'_j)\right)^2\}$, is equivalent to the minimization of the expected value of the error function [14]

$$E_N\{e^2(\bar{x}_{i,j}, \bar{x}'_j)\}, \tag{3.4}$$

where the expectation is computed over the number of training samples N.

Short and Fukunaga [14] notice that (3.4) can be minimized by either increasing N or designing a distance function $d(\bar{x}'_j, \cdot)$ which minimizes (3.4) in the classifiers. In a classification problem, a proper distance function is computed as [14]

$$d(\bar{x}'_j, \bar{x}_{i,j}) = \|\bar{\mu}(\bar{x}_{i,j}) - \bar{\mu}(\bar{x}'_j)\|_2^2, \tag{3.5}$$

where

$$\bar{\mu}(\bar{x}_{i,j}) = \left[\mu_1(\bar{x}_{i,j}), \ldots, \mu_c(\bar{x}_{i,j}), \ldots, \mu_C(\bar{x}_{i,j})\right],$$

$$\bar{\mu}(\bar{x}'_j) = \left[\mu_1(\bar{x}'_j), \ldots, \mu_c(\bar{x}'_j), \ldots, \mu_C(\bar{x}'_j)\right]$$

and $\|\cdot\|_2^2$ is the squared ℓ_2 norm, or Euclidean distance.

In a single classifier, (3.5) is computed in $F_j, \forall j$, using local approximations to *posterior* probabilities using training and test datasets [14]. Moreover, if the N-sample error is minimized on each different feature space $F_j, \forall j = 1, 2, \ldots, J$, then an average error over an ensemble of classifiers $\hat{E}_J\{E_N\{e^2(\bar{x}_{i,j}, \bar{x}'_j)\}\}$ which is defined as

$$\hat{E}_J\{E_N\{e^2(\bar{x}_{i,j}, \bar{x}'_j)\}\} = \frac{1}{J}\sum_{j=1}^{J} E_N\{e^2(\bar{x}_{i,j}, \bar{x}'_j)\} \tag{3.6}$$

is minimized by using

$$d(\bar{x}', \bar{x}_i) = \sum_{j=1}^{J}\sum_{c=1}^{C}(\mu_c(\bar{x}_{i,j}) - \mu_c(\bar{x}'_j))^2. \tag{3.7}$$

In this study, an approach to minimize (3.6) using (3.7) is employed in a hierarchical decision fusion algorithm. For this purpose, first *posterior* probabilities $\mu_c(\bar{x}_{i,j})$ are estimated using individual k-NN classifiers, which are called base-layer classifiers. Then the probability estimates $\mu_c(\bar{x}_{i,j})$ are concatenated to construct

$$\bar{\mu}(\bar{x}_i) = \left[\bar{\mu}(\bar{x}_{i,1}) \ldots \bar{\mu}(\bar{x}_{i,j}) \ldots \bar{\mu}(\bar{x}_{i,J})\right]$$

and

$$\bar{\mu}(\bar{x}') = \left[\bar{\mu}(\bar{x}'_1) \ldots \bar{\mu}(\bar{x}'_j) \ldots \bar{\mu}(\bar{x}'_J)\right],$$

for all training and test samples. Finally, classification is performed using $\bar{\mu}(\bar{x}')$ and $\bar{\mu}(\bar{x}_i), \forall i$, by a k-NN classifier, called meta-layer classifier, with

$$d(\bar{x}', \bar{x}_i) = \|\bar{\mu}(\bar{x}_i) - \bar{\mu}(\bar{x}')\|_2^2. \tag{3.8}$$

Note that (3.8) can be used for the minimization of the error difference in a feature space $F = F_1 \times F_2 \times \ldots \times F_J$. If $F = F_j$ for $j \in \{1, 2, \ldots, J\}$, then (3.8) is equal to (3.5). Therefore, distance learning problem proposed by Short and Fukunaga [14] is reformulated as a decision fusion problem. Then, the distance learning approach is employed using a hierarchical decision fusion algorithm called Fuzzy Stacked Generalization (FSG) [94] as described below.

3.4 Fuzzy Stacked Generalization

Given a training dataset $S = \{(s_i, y_i)\}_{i=1}^{N}$, each sample s_i is represented in J different feature spaces F_j, $j = 1, 2, \ldots, J$ by a feature vector $\bar{x}_{i,j} \in \mathbb{R}^{D_j}$ which is extracted by using the j^{th} feature extractor $FE_j, \forall j = 1, 2, \ldots, J$. Therefore, training datasets of base-layer classifiers employed on feature spaces $F_j, \forall j = 1, 2, \ldots, J$ can be represented by J different feature sets, $S_j = \{(\bar{x}_{i,j}, y_i)\}_{i=1}^{N}$.

At the base-layer, each feature vector extracted from the same sample is fed into an individual fuzzy k-NN classifier in order to estimate *posterior* probabilities using the class memberships as

$$\mu_c(\bar{x}_{i,j}) = \frac{\sum_{n=1}^{k} y_{l(n)} (\left\| \bar{x}_{i,j} - \bar{x}_{l(n),j} \right\|_2)^{-\frac{2}{\varphi-1}}}{\sum_{n=1}^{k} (\left\| \bar{x}_{i,j} - \bar{x}_{l(n),j} \right\|_2)^{-\frac{2}{\varphi-1}}}, \tag{3.9}$$

where $y_{l(k)}$ is the label of the k^{th}-nearest neighbor of $\bar{x}_{i,j}$ which is $\bar{x}_{l(k),j}$, and φ is the fuzzification parameter [110]. Each base-layer fuzzy k-NN classifier is trained and the membership vectors $\bar{\mu}(\bar{x}_{i,j})$ of each sample s_i is computed using leave-one-out cross validation. For this purpose, (3.9) is employed for each $(\bar{x}_{i,j}, y_i)$ using a validation set $S_j^{CV} = S_j - (\bar{x}_{i,j}, y_i)$, where $(\bar{x}_{l(k),j}, y_{l(k)}) \in S_j^{CV}$.

The class label of an unknown sample s_i is estimated by a base-layer classifier employed on F_j as

$$\hat{y}_{i,j} = \arg\max_{\omega_c} (\bar{\mu}(\bar{x}_{i,j})).$$

The training performance of the j^{th} base-layer classifier is computed as,

$$Perf_j^{tr} = \frac{1}{N} \sum_{i=1}^{N} \delta_{\hat{y}_{i,j}}(S_j), \tag{3.10}$$

where

$$\delta_{\hat{y}_{i,j}}(S_j) = \begin{cases} 1, & \text{if } y_i \equiv \hat{y}_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

is the Kronecker delta which takes the value 1 when the j^{th} base-layer classifier correctly classifies a sample $s_i \in S_j$ such that $y_i \equiv \hat{y}_{i,j}$.

When a set of test samples $S_j^{te} = \{s_i'\}_{i=1}^{N'}$ is received, the feature vectors $\{\bar{x}_{i,j}'\}_{i=1}^{N'}$ of the samples are extracted by each FE_j. Then, posterior probability $\mu_c(\bar{x}_{i,j}')$ of each test sample s_i', $i = 1, 2, \ldots, N'$ is estimated using the training datasets S_j by each base-layer k-NN classifier at each F_j, $\forall j = 1, 2, \ldots, J$.

If a set of labels of test samples, $\{y_i'\}_{i=1}^{N'}$, is available, then the test performance is computed as

$$Perf_j^{te} = \frac{1}{N'} \sum_{i=1}^{N'} \delta_{\hat{y}_{i,j}'}(S_j^{te}).$$

The output space of each base-layer classifier is spanned by the membership vectors of each feature vector $\bar{x}_{i,j}$. It should be noted that the membership vectors satisfy

$$\sum_{c=1}^{C} \mu_c(\bar{x}_{i,j}) = 1.$$

This equation aligns each sample on the surface of a simplex at the output space of a base-layer classifier, which is called a *Decision Space* of that classifier. Therefore, base-layer classifiers can be considered as transformations which map the input feature space of any dimension into a point on a simplex in a C (number of classes) dimensional decision space (for $C = 2$, the simplex is reduced to a line).

Class-membership vectors obtained at the output of each classifier are concatenated to construct a feature space called *Fusion Space* for a meta-layer classifier. The fusion space consists of CJ dimensional feature vectors $\bar{\mu}(\bar{x}_i)$ and $\bar{\mu}(\bar{x}'_i)$ which form the training dataset

$$S_{meta} = \{(\bar{\mu}(\bar{x}_i), y_i)\}_{i=1}^{N}$$

and the test dataset

$$S'_{meta} = \{\bar{\mu}(\bar{x}'_i)\}_{i=1}^{N'}$$

for the meta-layer classifier. Note that

$$\sum_{j=1}^{J}\sum_{c=1}^{C}\mu_c(\bar{x}_{i,j}) = J \text{ and } \sum_{j=1}^{J}\sum_{c=1}^{C}\mu_c(\bar{x}'_{i,j}) = J.$$

Finally, a meta-layer fuzzy k-NN classifier is employed to classify the test samples using their feature vectors in S'_{meta} with the feature vectors of training samples in S_{meta}. Meta-layer training and test performance is computed as

$$Perf^{tr}_{meta} = \frac{1}{N}\sum_{i=1}^{N}\delta_{\hat{y}_{i,meta}}(S_{meta})$$

and

$$Perf^{te}_{meta} = \frac{1}{N'}\sum_{i=1}^{N'}\delta_{\hat{y}'_{i,meta}}(S'_{meta}),$$

respectively. An algorithmic description of the FSG is given in Algorithm 1.

input : Training set $S = \{(s_i, y_i)\}_{i=1}^{N}$, test set $S_j^{te} = \{s'_i\}_{i=1}^{N'}$ and J feature extractors FE_j, $\forall j = 1, 2, \ldots, J$.
output: Predicted class labels of the test samples $\{\hat{y}'_i\}_{i=1}^{N'}$.
foreach $j = 1, 2, \ldots, J$ **do**
1 Extract features $\{\bar{x}_{i,j}\}_{i=1}^{N}$ and $\{\bar{x}'_{i,j}\}_{i=1}^{N'}$ using FE_j;
2 Compute $\{\bar{\mu}(\bar{x}_{i,j})\}_{i=1}^{N}$ and $\{\bar{\mu}(\bar{x}'_{i,j})\}_{i=1}^{N'}$ using (3.9);
end
3 Construct $S_{meta} \coloneqq \{(\bar{\mu}(\bar{x}_i), y_i)\}_{i=1}^{N}$ and $S'_{meta} \coloneqq \{\bar{\mu}(\bar{x}'_i)\}_{i=1}^{N'}$;
4 Employ meta-layer classification using S_{meta} and S'_{meta} to predict $\{\hat{y}'_i\}_{i=1}^{N'}$;

Algorithm 1: Fuzzy Stacked Generalization.

Notice that the Fuzzy Stacked Generalization fuses the posterior probabilities obtained at the output of base-layer classifiers under the fusion space and then updates these posterior probabilities at the output of the meta-layer classifier.

3.5 Minimization of Large-sample Classification Error in the FSG

Cover and Hart [13] state that one of the conditions required to converge $\epsilon(\bar{x}'_j)$ to $e*$ is to estimate posterior probabilities with complete certainty or uncertainty. For instance, in a two

class classification problem, if $\epsilon(\bar{x}'_j) = 0, 1$ or $\frac{1}{2}$, then $\epsilon(\bar{x}'_j)$ is equal to $e*$. If $\epsilon(\bar{x}'_j) = 1$, then the classification performance is 0, i.e. all the samples are misclassified. If $\epsilon(\bar{x}'_j) = \frac{1}{2}$, then all the samples are correctly classified or misclassified by chance. Therefore, we focus on the condition $\epsilon(\bar{x}'_j) = 0$, where all the samples are correctly classified. In this section, this condition is analyzed using a weighted decision algorithm and two sample selection algorithms.

In order to satisfy the conditions stated by Cover and Hart [13], one must assure that the samples belonging to the same class reside in the same Voronoi region [4] in the fusion space. For this purpose, first the contribution of each sample to the classification performance of the meta-layer classifier should be measured. This task is achieved by introducing a new concept called Decision Margin which will defined in the following subsection.

3.5.1 Decision Margin

The relationship between the contribution of a training and a test sample, s_i and s'_i, to the classification performances of the base-layer classifiers and that of the meta-layer classifier is analyzed by first defining the target *posterior* probability values of s_i and s'_i as

$$\mu_c(s_i, j) = \begin{cases} 1, \text{ if } y_i \in \omega_c \\ 0, \text{ otherwise} \end{cases} \quad \text{and} \quad \mu_c(s'_i, j) = \begin{cases} 1, \text{ if } y'_i \in \omega_c \\ 0, \text{ otherwise} \end{cases}, \tag{3.12}$$

$$\bar{\mu}(s_i, j) = [\mu_1(s_i, j), \ldots, \mu_C(s_i, j)] \text{ and } \bar{\mu}(s'_i, j) = [\mu_1(s'_i, j), \ldots, \mu_C(s'_i, j)], \tag{3.13}$$

$$\bar{\mu}(s_i) = [\bar{\mu}(s_i, 1), \ldots, \bar{\mu}(s_i, J)] \text{ and } \bar{\mu}(s'_i) = [\bar{\mu}(s'_i, 1), \ldots, \bar{\mu}(s'_i, J)], \tag{3.14}$$

respectively. Then, we define a training and a test error function to measure the classifiability of s_i and s'_i as

$$e(s_i, j) = \sum_{c=1}^{C} \parallel \mu_c(s_i, j) - \mu_c(\bar{x}_{i,j}) \parallel_2^2 \text{ and } e(s'_i, j) = \sum_{c=1}^{C} \parallel \mu_c(s'_i, j) - \mu_c(\bar{x}'_{i,j}) \parallel_2^2 . \tag{3.15}$$

Let us denote $E_N\{e^2(s_i, j)\}$ and $E_{N'}\{e^2(s'_i, j)\}$ as the N-sample training and test errors, respectively. Similar to the strategy for defining the distance function (3.5) to minimize the error (3.4) in Section 3.3, two distance functions are proposed to minimize $E_N\{e^2(s_i, j)\}$ and $E_{N'}\{e^2(s'_i, j)\}$ as

$$\rho(s_i, j) = \|\bar{\mu}(s_i, j) - \bar{\mu}(\bar{x}_{i,j})\|_2 \text{ and } \rho(s'_i, j) = \|\bar{\mu}(s'_i, j) - \bar{\mu}(\bar{x}'_{i,j})\|_2. \tag{3.16}$$

$\rho(s_i, j)$ is called *Decision Margin* of the training sample s_i and $\rho(s'_i, j)$ is called *Decision Margin* of the test sample s'_i in a feature space F_j.

By the definition given in (3.12), decision margin provides information on *the magnitude* with which a sample is classified in a feature space F_j. For instance, in a two class classification problem with $C = 2$, if a sample s_i with $y_i \in \omega_1$ is correctly classified with high probability in F_j such that $\mu_1(\bar{x}_{i,j}) = 1$, then $\rho(s_i, j) = 0$. On the other hand, if the sample s_i is classified *by chance* i.e. $\mu_1(\bar{x}_{i,j}) = \mu_2(\bar{x}_{i,j})(= 0.5)$, then $\rho(s_i, j) = \sqrt[2]{\frac{1}{2}} \approx 0.7$. If the sample is misclassified, then $\rho(s_i, j)$ takes the maximum value for this example, which is $\rho(s_i, j) = \sqrt[2]{2} \approx 1.41$.

The information about the classifiability of the samples at the meta-layer is obtained using following error functions;

$$e(s_i) = \sum_{j=1}^{J} e(s_i, j) \text{ and } e(s'_i) = \sum_{j=1}^{J} e(s'_i, j), \tag{3.17}$$

for training and test samples, respectively.

Note that the membership values of all the samples lie on the surface of a simplex in the C-dimensional output space of each base-layer classifier. In practice, the entry of the vector $\bar{\mu}(\bar{x}'_{i,j})$ with the highest membership value shows the estimated class label $\hat{y}_{i,j}$ in F_j, and the membership vector of a correctly classified sample is expected to accumulate around the *correct* vertex of the simplex. Concatenation operation creates a CJ-dimensional fusion space at the input of the meta-layer classifier in which the membership values lie on the CJ-dimensional simplex. The membership values of the correctly classified samples, this time, form even a more compact cluster around each vertex of the simplex, and misclassified samples are scattered all over the surface. If a sample s_i is correctly classified by at least one base-layer classifier in F_j with $\rho(s_i, j)$, then the corresponding membership value will be close to the *correct* vertex of the simplex by a magnitude of $\rho(s_i, j)$, and will contribute to increase the performance of overall FSG. Let us depict the above fact by an example.

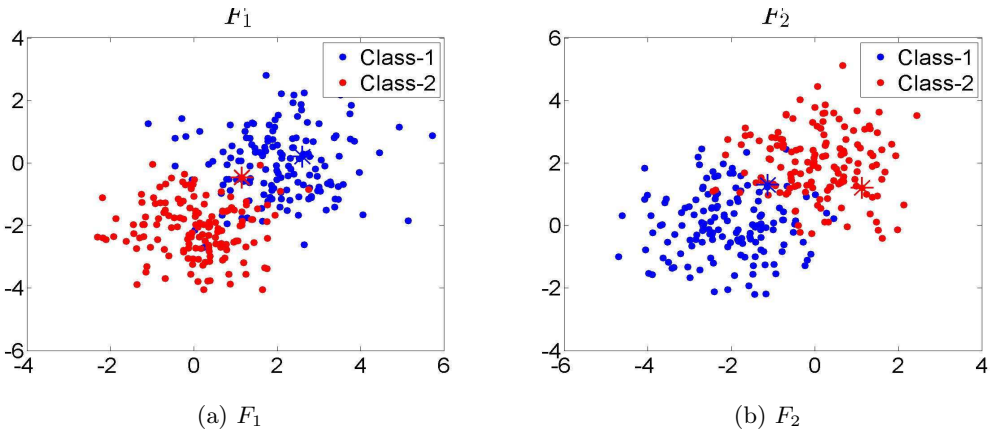
Example: Consider a synthetic dataset, consisting of $C = 2$ classes each of which consists of 250 samples represented in $J = 2$ distinct feature spaces. In the base-layer feature spaces shown in Figure 3.1, the classes have Gaussian distribution with substantial overlaps where the mean value and covariance matrices are

$$M_1 = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix},\ T_1 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \text{ and } M_2 = \begin{pmatrix} -2 & 0 \\ 2 & 2 \end{pmatrix},\ T_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

for the first and the second feature spaces, respectively. The features of the samples from the first class are represented by blue dots and that of the second class are represented by red dots. Features of two randomly selected samples, which are misclassified by one of the base-layer classifiers and correctly classified by the meta-layer classifier, are shown by star markers. In the feature spaces, each of the training samples is correctly classified by at least one base-layer fuzzy k-NN classifier with $k = 3$. The classification performances of the base-layer classifiers are 91% and 92% respectively. The classification performance of the FSG is 96%.

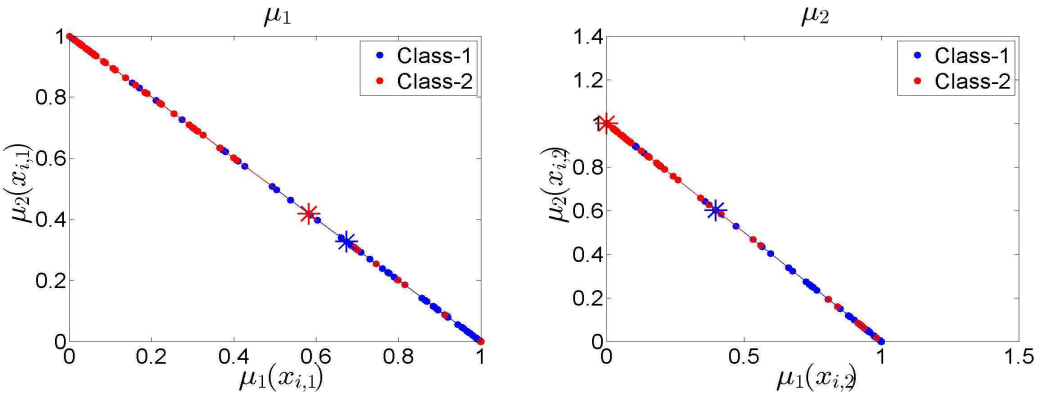
The membership values lie on a line at the output spaces of two base-layer classifiers, as depicted in Figure 3.2. In these figures, the decisions of the classifiers are also depicted for individual samples. For instance, the sample marked with red star s_1 is misclassified by the first classifier as shown in Figure 3.2.a with $\rho(s_1, 1) = 0.823$, but correctly classified by the second classifier with $\rho(s_1, 2) = 0$ as shown in Figure 3.2.b. In addition, the feature of the sample marked with blue star s_2 is correctly classified by the first classifier as shown in Figure 3.2.a with $\rho(s_2, 1) = 0.463$, but misclassified by the second classifier with $\rho(s_2, 2) = 0.851$ as shown in Figure 3.2.b.

The concatenation operation creates a 4 (2∗2) dimensional fusion space at the meta-layer input feature space. In order to visualize the distribution of the samples, four different subspaces, each of which is a 3-dimensional Euclidean space, are selected. Figure 3.3 displays different combinations of the subspaces and the membership vectors obtained from each classifier. Notice that the concatenation operation forms planes in these subspaces accumulating the correctly classified samples around the edges and the vertices. Therefore, features of the



(a) F_1 (b) F_2

Figure 3.1: Feature vectors in (a) F_1 and (b) F_2. Features of two randomly selected samples are indicated by (*) to follow them at the decision spaces of base-layer classifiers and the fusion space.



(a) (b)

Figure 3.2: Membership vectors obtained at the output of base-layer classifiers: (a) Classifier 1 and (b) Classifier 2. The locations of the features of randomly selected samples of Figure 3.1 are indicated by (*), at each simplex.

samples which are correctly classified by at least one base-layer classifier are located close to one of the *correct* vertices, or edges. This fact is depicted in Figure 3.3, where the feature of the sample indicated by red star is located close to the edges of the second class in Figure 3.3.b, c, d. On the other hand, the feature of the sample indicated by blue star is located close to the edges of the first class in Figure 3.3.a, c, d. Both of these samples are correctly labeled by the meta-layer fuzzy k-NN classifier.

Since the decision margin $\rho(s_i, j)$ and the error $e(s_i, j)$ of a sample s_i is linearly related, there is also a linear relationship between $\rho(s_i, j)$ and $e(s_i)$. Therefore, if a sample or a group of samples in the training data is correctly classified by the base-layer classifiers with minimum decision margin, then these sample groups contribute to improve the overall performance of the FSG. Otherwise, the samples which are not correctly classified by any of the base-layer classifiers with the highest decision margin become spurious, and distort the feature space at the input of the meta-layer classifier. This observation is consistent with the works of Wolpert

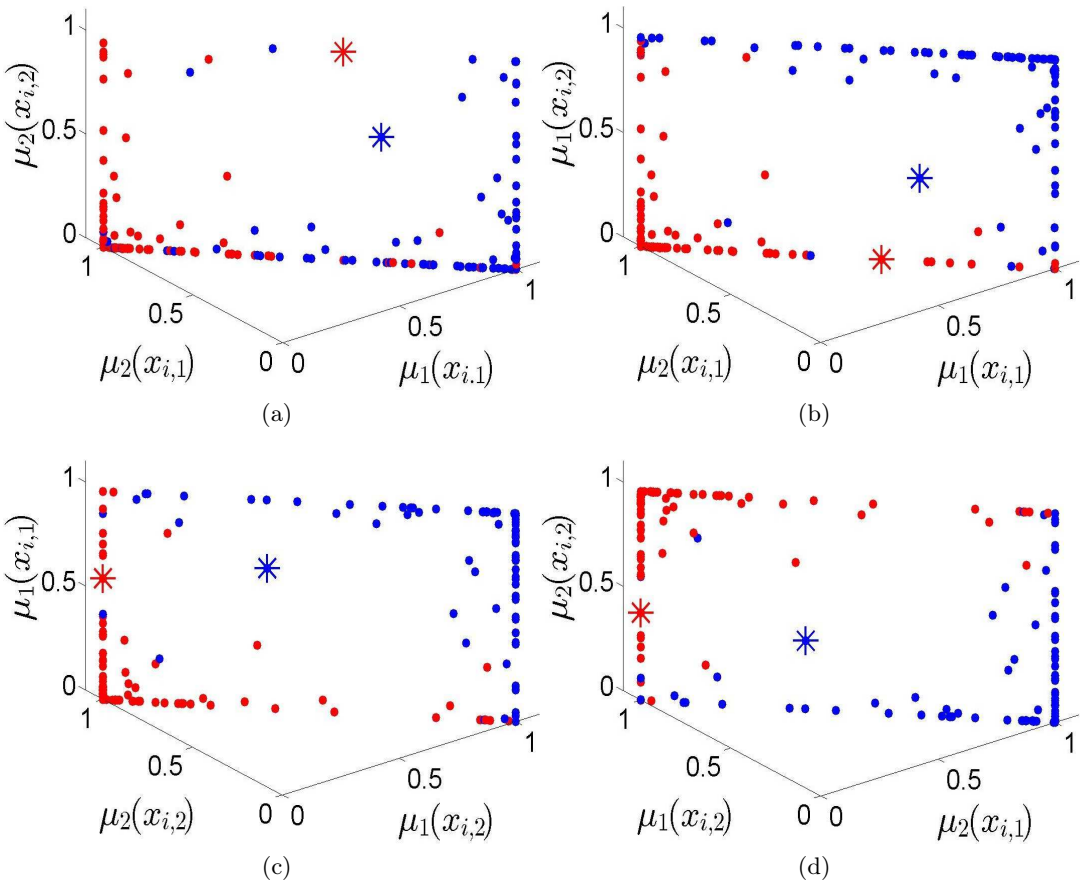


Figure 3.3: The relationships among (a) $\mu_1(x_i,1)$, $\mu_2(x_i,1)$, $\mu_2(x_i,2)$, (b) $\mu_1(x_i,1)$, $\mu_2(x_i,1)$, $\mu_1(x_i,2)$, (c) $\mu_1(x_i,2), \mu_(x_i,2), \mu_1(x_i,1)$, and (d) $\mu_2(x_i,1)$, $\mu_1(x,2)$, $\mu_2(x_i,2)$, are visualized. The locations of the features of randomly selected samples of Figure 3.1 are indicated by (*) at the subspaces of the fusion space.

[84], Ting and Witten [85] which claim that the individual classifiers can identify different parts of the feature space. Therefore, base-layer feature spaces, classifiers and decision fusion methods should be designed in such a way that feature vectors of samples are shared among the base-layer classifiers to recognize the samples in the training set, and cover the entire feature space. If this is not possible, it may be wise to eliminate the spurious samples which spoil the separability of the fusion space.

In this work, decision margin is used as a criterion to measure the contribution of each sample to base-layer and meta-layer classification performances. For this purpose, decision fusion method of FSG is redesigned by taking into account the decision margins of samples. In this approach, decisions of base-layer classifiers are fused in order to maximize the contributions of samples to meta-layer classification performance by minimizing their decision margins. Therefore, decision of each base-layer classifier is weighted according to decision margins of samples computed in the classifier. Then, the weights which minimize decision margins of samples and the meta-layer classification error are computed by solving a convex optimization problem using a weighted decision fusion algorithm called Decision Fusion using Decision Margin in the next subsection.

3.5.2 Weighted Decision Fusion using Decision Margin in the FSG

In the previous sections, it is shown that the training and test errors are minimized when the decision margins of the training and test samples are minimized. Therefore, the decision fusion problem of the FSG is defined as the decision margin minimization problem in this section.

First a *weighted decision margin* is defined as

$$\rho(s_i, j)_{\bar{w}_j} = \|\bar{\mu}(\bar{x}_{i,j})\bar{w}_j - \bar{\mu}(s_i, j)\|_2, \tag{3.18}$$

where $\bar{w}_j \in \mathbb{R}^C$ is a weight vector used for the weighted fusion of the classifier decisions. Note that decision margin of a sample s computed by a classifier in a feature space F_j gives information about the decision of that classifier on the class membership of s and the contribution of the decision of the base-layer classifier on s to the decision of a meta-layer classifier. Therefore, the weight vectors are employed to increase the *certainty* of the decision of the meta-layer classifier on the samples using the information on the relationship between the classifiability of the samples and the classification performances of base-layer and meta-layer classifiers.

The analyses given in the previous sections show that the minimization of decision margins of samples enables us to minimize large-sample error of the meta-layer k-NN classifier. In addition, decision margins can be used to accumulate the samples belonging to the same classes towards the vertices of polytopes of decision vectors in the fusion space which represent the target (class) locations of the feature vectors of the samples that are defined by $\bar{\mu}(s_i, j)$ of the samples s_i, $\forall i = 1, 2, \ldots, N$ and $\forall j = 1, 2, \ldots, J$.

Then, the following weighted decision margin minimization problem is defined for the fusion of the classifier decisions;

$$\text{minimize} \qquad \sum_{i=1}^{N} \sum_{j=1}^{J} \rho(s_i, j)^2_{\bar{w}_j} + \lambda \sum_{j=1}^{J} \|\bar{w}_j\|_2, \tag{3.19}$$

where the ℓ_2 norm regularization parameter λ is used for inducing sparsity on the weight vectors used for classifier decision fusion. Therefore, the number of classifiers whose decisions will not be considered in the decision fusion increases as λ increases. Note that (3.19) has been studied as the *Group Lasso* problem in the statistical data analysis and learning problems [111].

This problem is modeled as a convex optimization problem in the following form

$$\begin{aligned} &\text{minimize} && \lambda \sum_{j=1}^{J} \|\bar{w}_j\|_2, \\ &\text{subject to} && \bar{\mu}(\bar{x}, j)\bar{w}_j - \bar{\mu}(s_i, j) = 0, \qquad \forall j = 1, 2, \ldots, J, \\ & && \qquad\qquad\qquad\qquad\quad \forall i = 1, 2, \ldots, N. \end{aligned} \tag{3.20}$$

The Lagrangian of (3.20) is defined as

$$L(\bar{w}_j, \bar{\alpha}) = \lambda \sum_{j=1}^{J} \|\bar{w}_j\|_2 + \bar{\alpha}^\top \Big(\sum_{i=1}^{N} \sum_{j=1}^{J} \big(\bar{\mu}(\bar{x}, j)\bar{w}_j - \bar{\mu}(s_i, j)\big)\Big), \tag{3.21}$$

where $\bar{\alpha}$ is a vector of optimization dual variables and $\bar{\alpha}^\top$ is the transpose of $\bar{\alpha}$. The dual function is defined as

$$\Upsilon(\bar{\alpha}) = \inf_{\bar{w}_j} L(\bar{w}_j, \bar{\alpha}). \tag{3.22}$$

Then the dual problem is defined as

$$\text{maximize } \Upsilon(\bar{\alpha}). \tag{3.23}$$

An optimal solution for (3.20) is $\bar{w}_j^*$ and for (3.23) is $\bar{\alpha}^*$. When the dual problem (3.23) is solved, the primal solution $\bar{w}_j^*$ can be obtained using

$$\bar{w}_j^* = \arg\min_{\bar{w}_j} L(\bar{w}_j, \bar{\alpha}*). \tag{3.24}$$

In the *dual ascent method* [112], the the dual problem (3.23) is solved using gradient ascent on $\Upsilon(\bar{\alpha})$ by computing the gradient $\nabla\Upsilon(\bar{\alpha})$, assuming that Υ is differentiable. In the algorithm, first a *local* primal solution is computed as

$$\bar{w}_j^{t+1} \coloneqq \arg\min_{\bar{w}_j} L(\bar{w}_j, \bar{\alpha}_t). \tag{3.25}$$

Then, using the following approximation to the gradient

$$\nabla\Upsilon(\bar{\alpha}) \approx \bar{\mu}(\bar{x}, j)\bar{w}_j^{t+1} - \bar{\mu}(s_i, j), \tag{3.26}$$

the following dual variable update step is employed

$$\bar{\alpha}_{t+1} \coloneqq \bar{\alpha}_t + \upsilon^t\Big(\sum_{i=1}^{N}\sum_{j=1}^{J}\bar{\mu}(\bar{x}, j)\bar{w}_j^{t+1} - \bar{\mu}(s_i, j)\Big), \tag{3.27}$$

where υ^t is a step size which can be defined at each iteration t [112].

Since the decision margin problem (3.20) is separable with respect to the classifier decisions, (3.25) can be solved at each local processor (classifier) independently and in parallel. Then, the local solutions $\bar{w}_j^{t+1}$ are *gathered* or *collected* at the master-processor (meta-classifier) to compute the residual

$$\upsilon^t\Big(\sum_{i=1}^{N}\sum_{j=1}^{J}\bar{\mu}(\bar{x}, j)\bar{w}_j^{t+1} - \bar{\mu}(s_i, j)\Big).$$

Finally, the *global* dual solution of (3.27) is *distributed* or *broadcasted* to the local processors to update the local solutions in the next epoch.

In order to relax the convexity and finiteness requirements of (3.19), the problem is written as

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{J}\|\bar{\mu}(s, j) - \bar{\gamma}_j\|_2^2 + \lambda\sum_{j=1}^{J}\|\bar{w}_j\|_2, \\ \text{subject to} \quad & \bar{\mu}(\bar{x}, j)\bar{w}_j - \bar{\gamma}_j = 0, \forall j = 1, 2, \ldots, J, \end{aligned} \tag{3.28}$$

where $\bar{\mu}(s, j) = [\bar{\mu}(s_1, j)^\top, \ldots, \bar{\mu}(s_N, j)^\top]^\top$, $\bar{\mu}(\bar{x}, j) = [\bar{\mu}(x_{1,j})^\top, \ldots, \bar{\mu}(x_{N,j})^\top]^\top$, $\bar{\gamma}_j$ is a vector of optimization variables. Then, the *augmented* Lagrangian is written as [112]

$$L_\theta(w_j, \bar{\alpha}) = \sum_{j=1}^{J}\|\bar{\mu}(s, j) - \bar{\gamma}_j\|_2^2 + \lambda\sum_{j=1}^{J}\|\bar{w}_j\|_2 + \bar{\alpha}^\top\Big(\sum_{j=1}^{J}\bar{\mu}(\bar{x}, j)\bar{w}_j - \bar{\gamma}_j\Big) + \frac{\theta}{2}\|\bar{\mu}(\bar{x}, j)\bar{w}_j - \bar{\gamma}_j\|_2^2, \tag{3.29}$$

where $\theta > 0$ is a penalty parameter .

Then, (3.29) is solved in the following three steps:

1. $\bar{w}_j$ update step:

$$\bar{w}_j^{t+1} := \underset{\bar{w}_j}{\operatorname{argmin}}\Big(\frac{\theta}{2}\|\bar{\mu}(\bar{x},j)\bar{w}_j - \bar{\mu}(\bar{x},j)\bar{w}_j^t + \frac{1}{J}\sum_{j=1}^{J}\big(\bar{\mu}(\bar{x},j)\bar{w}^t - \bar{\gamma}_j\big) + \bar{u}^t\|_2^2 + \lambda\|\bar{w}_j\|_2\Big), \quad (3.30)$$

where $\bar{u}^t = \frac{1}{\theta}\bar{\alpha}_t$ is a vector of scaled dual optimization variables computed at an iteration t.

2. $\tilde{\bar{\gamma}}$ update step:

$$\tilde{\bar{\gamma}}^{t+1} := \frac{1}{J+\theta}\Big(\bar{\mu}(s,j) + \theta\frac{1}{J}\sum_{j=1}^{J}\big(\bar{\mu}(\bar{x},j)\bar{w}_j^{t+1}\big) + \theta\bar{u}^t\Big). \quad (3.31)$$

3. $\bar{u}$ update step:

$$\bar{u}^{t+1} := \bar{u}^t + \frac{1}{J}\sum_{j=1}^{J}\big(\bar{\mu}(\bar{x},j)\bar{w}_j^{t+1}\big) - \tilde{\bar{\gamma}}^{t+1}. \quad (3.32)$$

In the first step of the algorithm, (3.30) is solved at each classifier individually. Then, the local solutions of (3.30) are gathered at the meta-layer classifier to update an average $\tilde{\bar{\gamma}}$ by computing (3.31). Finally, $\tilde{\bar{\gamma}}^{t+1}$ is distributed among the classifier to update $\bar{u}^t$ using (3.32). The algorithm has been referred as Alternating Direction Method of Multipliers (ADMM) [112] in the literature.

input : Training set $S = \{(s_i, y_i)\}_{i=1}^N$, feature extractors FE_j, $j = 1, 2, \ldots, J$, the parameters λ, θ, $\bar{\alpha}$ and algorithm termination time T.
output: Decision weights $\{\bar{w}_j\}_{j=1}^J$.

1 $t \leftarrow \mathbf{1}$ and $\bar{\alpha}_t \leftarrow \bar{\alpha}$;
2 $\bar{w}_j(t) \leftarrow 0$, $\forall j = 1, 2, \ldots, J$;
foreach $j = 1, 2, \ldots, J$ **do**
3 Extract features $\{\bar{x}_{i,j}\}_{i=1}^N$ using FE_j;
4 Compute $\{\bar{\mu}(\bar{x}_{i,j})\}_{i=1}^N$ using (3.9);
end
repeat
5 Update weights $\bar{w}_j^{t+1}$ using (3.30), $\forall j = 1, 2, \ldots, J$;
6 Gather weights $\bar{w}_j^{t+1}$ at the meta-classifier to update $\tilde{\bar{\gamma}}^{t+1}$ using (3.31) ;
7 Update $\bar{u}^{t+1}$ using (3.32), and distribute $\bar{u}^{t+1}$ to the base-layer classifiers $\forall j = 1, 2, \ldots, J$;
8 $t \leftarrow t + 1$;
until $t \leq T$;
9 $\bar{w}_j \leftarrow \bar{w}_j(T)$, $\forall j = 1, 2, \ldots, J$;

Algorithm 2: Decision Fusion using weighted Decision Margin minimization in the Fuzzy Stacked Generalization.

An algorithmic description of the weighted decision fusion algorithm for the FSG is given in Algorithm 2. In the first and second steps of the algorithm, variables are initialized. Features are extracted from the samples in the third step, and class membership vectors which represent the decisions of the base-layer classifiers are computed in the fourth step. In each base-layer classifier employed on each feature space F_j, posterior probability vectors are estimated (step 4). At an epoch t of the algorithm, first weight vectors $\hat{w}_j(t)$ are *locally* computed in each

base-layer individual classifier (step 5). Then, locally computed weights $\bar{w}_j^{t+1}$ are gathered at the meta-classifier to compute the *global* solution $\tilde{\bar{\gamma}}^{t+1}$ using (3.31) in step 6, which is then distributed to the base-layer classifiers to update $\bar{u}^t$ in step 7. Algorithm epoch is updated in step 8 and the algorithm is iterated until a given termination time T. The computed solution $\hat{w}_j(T)$ is returned as the final solution in step 9.

At the meta-layer, decision vectors of base-layer classifiers are updated using weight vectors $\bar{w}_j$. Then, the updated decision vectors are fused for classification at the meta-layer classifier.

3.5.3 Sample Selection for Decision Fusion in the FSG

Recall that the condition stated by Cover and Hart [13] and analyzed in the previous subsections, which is the estimation of posterior probabilities with complete certainty or uncertainty, should be satisfied almost everywhere, i.e. at each feature vector of each sample in each feature space.

Following the sample selection approach of Wilson [35], this condition is satisfied by eliminating the samples which are not correctly classified by at least one base-layer classifier from the training dataset in this subsection.

Sample selection methods have been used by Ozay and Yarman Vural [94, 95, 70, 113, 114, 115, 116] for base-layer feature space design by eliminating the samples from both training and test datasets. In this work, two sample selection algorithms are proposed to minimize the difference between large-sample error of k-NN and Bayes Error.

In order to determine whether a sample is correctly classified by at least one base-layer classifier or not, first a misclassification index is defined for each sample as

$$mc(s_i) = \begin{cases} 1, \; if \left(\bigvee_{j=1}^{J} \delta_{\hat{y}_{i,j}}(S_j) \right) \equiv 0 \\ 0, \; otherwise \end{cases}, \tag{3.33}$$

where

$$\delta_{\hat{y}_{i,j}}(S_j) = \begin{cases} 1, \; if \; y_i \equiv \hat{y}_{i,j} \\ 0, \; otherwise \end{cases} \tag{3.34}$$

Therefore, if a sample is correctly classified by at least one base-layer classifier, then its misclassification index is 0, otherwise it is 1. In addition, a set of samples which are misclassified by all the base-layer classifiers is defined as $MC = \{s_i : mc(s_i) \equiv 1\}$.

In the first sample selection algorithm (SS-1) given in Algorithm 3, the samples belonging to the misclassified sample set are removed from the meta-layer input training dataset in order to obtain $\hat{S}_{meta}$, which is a new set of meta-layer features with the associated class labels of the samples that are correctly classified by at least one base-layer classifier. Then, $\hat{S}_{meta}$ is used to label the meta-layer features of test samples.

In Algorithm 3, a new set of membership vectors in S'_{meta} are computed using the membership vectors in S_{meta}. However, this approach may result in some information loss about the eliminated samples in S'_{meta} and the statistical stability between meta-layer input training and test datasets may be damaged. In order to avoid this undesired loss and assure the

input : Training set $S = \{(s_i, y_i)\}_{i=1}^{N}$, test set $S_j^{te} = \{s_i'\}_{i=1}^{N'}$ and feature extractors $FE_j, \forall j = 1, 2, \ldots, J$.

output: Predicted class labels of the test samples $\{\hat{y}_i'\}_{i=1}^{N'}$.

foreach $j = 1, 2, \ldots, J$ **do**

1 Extract features $\{\bar{x}_{i,j}\}_{i=1}^{N}$ and $\{\bar{x}_{i,j}'\}_{i=1}^{N}$ using FE_j;

2 Compute $\{\bar{\mu}(\bar{x}_{i,j})\}_{i=1}^{N}$ and $\{\bar{\mu}(\bar{x}_{i,j}')\}_{i=1}^{N}$ using (3.9);

end

3 Construct MC ;

4 Construct $S'_{meta} \coloneqq \{\bar{\mu}(\bar{x}_i')\}_{i=1}^{N'}$;

5 Eliminate spurious training samples from meta-layer input training dataset by constructing

$$\hat{S}_{meta} \coloneqq \{(\bar{\mu}(\bar{x}_i), y_i) : s_i \notin MC \text{ and } s_i \in S, \forall i = 1, 2, \ldots, N\}.$$

6 Employ meta-layer classification using S'_{meta} and $\hat{S}_{meta}$ to predict $\{\hat{y}_i'\}_{i=1}^{N'}$;

Algorithm 3: Spurious training data elimination from meta-layer input training dataset (SS-1).

statistical stability of the samples at the meta-layer, the membership vectors of test samples are also updated at $\hat{S}_{meta}$ in Algorithm 4 as follows.

input : Training set $S = \{(s_i, y_i)\}_{i=1}^{N}$, test set $S_j^{te} = \{s_i'\}_{i=1}^{N'}$ and feature extractors $FE_j, \forall j = 1, 2, \ldots, J$.

output: Predicted class labels of the test samples $\{\hat{y}_i'\}_{i=1}^{N'}$.

1 Construct MC ;

2 Construct $\hat{S} \coloneqq \{(s_i, y_i) : s_i \notin MC \text{ and } s_i \in S, \forall i = 1, 2, \ldots, N\}$;

3 Construct S_{meta} and S'_{meta} using $\hat{S}$ as the base-layer input training dataset;

4 Employ meta-layer classification using S'_{meta} and S_{meta} to predict $\{\hat{y}_i'\}_{i=1}^{N'}$;

Algorithm 4: Spurious training data elimination from base-layer input training dataset (SS-2).

In Algorithm 4 (SS-2), first the samples belonging to MC are eliminated from the base-layer feature training set in order to construct a new base-layer training set $\hat{S}$. Then, the base-layer classifiers are re-trained and the membership vectors of both the training and test samples are computed using $\hat{S}$. Finally, the test samples are classified at the meta-layer.

The proposed algorithms have been analyzed on synthetic and benchmark datasets in Section 3.7. Computational complexity of the algorithms are given in the next section.

3.6 Computational Complexity of the Algorithms

In the analysis of the computational complexities of the proposed decision fusion algorithms, computational complexities of feature extraction algorithms are ignored assuming that the feature sets are already computed and given.

The computational complexity of the Fuzzy Stacked Generalization algorithm is dominated by the number of samples. The computational complexity of a base-layer k-NN classifier is

$O(ND_j)$. If each base-layer classifier is implemented by an individual processor in parallel, then the computational complexity of base-layer classification process is $O(N\tilde{D})$, where $\tilde{D} = max\{D_j\}_{j=1}^{J}$. In addition the computational complexity of a meta-layer classifier which employs fuzzy k-nn is $O(NJC)$. Therefore, the computational complexity of the FSG is $O(N(\tilde{D} + JC))$.

In Algorithm 2 which employs weighted decision fusion in FSG, the computational complexity of the computation of membership decision vectors is $O(N\tilde{D})$ as mentioned above. In addition, if the variable transmission time among the processors is ignored, the computational complexity of the weight vector computation steps is dominated by the weight update step. In this step, a regularized regression problem is solved at each base-layer classifier. Note that this problem can be approximated as a Tikhonov regularization problem in which Cholesky decomposition can be employed on $\bar{\mu}(\bar{x}, j)^{\top}\bar{\mu}(\bar{x}, j)$ [112]. Therefore, the computational complexity of this operation employed by a base-layer classifier is $O(ND_j)$ for each step $t \leq T$. Since matrix-vector multiplications are employed in the other update steps, their computational complexities are also $O(ND_j)$ in each base-layer classifier. If the computation is terminated after T iterations, then the complexity of the overall computation process is $O(TN\tilde{D})$.

Computational complexities of the sample selection algorithms are affected by the size $|MC|$ of the MC set. In both of the algorithms, Algorithm 3 and 4, membership vectors are computed by the base-layer classifiers with $O(N\tilde{D})$. In order to employ the sample selection in a linear time complexity, first each classifier constructs a $0-1$ predicate matrix whose elements are the values of the Dirac functions (3.34), in parallel. Then, MC can be constructed using a search algorithm on the concatenated predicate matrices, whose complexity is $O(N)$.

In Algorithm 3, the samples are eliminated from meta-layer input training dataset. Therefore, the computational complexity of the meta-layer classifier is $O(N - |MC|)$. In Algorithm 4, the samples are eliminated from base-layer input training dataset. Then, the classifiers are re-trained with $O((N - |MC|)\tilde{D})$ in the base-layer, and with $O((N - |MC|)JC)$ in the meta-layer.

3.7 Experimental Analysis of the Decision Fusion Algorithms

In this section, two sets of experiments are performed to analyze the behavior of the suggested supervised decision fusion algorithms. First, the proposed algorithms are tested on the synthetic datasets following the comments of Cover and Hart [13] for the analysis of the relationship between the class conditional densities of the datasets and the performance of the $k = 1$ nearest neighbor classification algorithm, and the metric learning method of Hastie and Tibshirani [117] to minimize the difference between the N-sample and large-sample classification error. Second, benchmark pattern classification datasets such as Breast Cancer, Diabetis, Flare Solar, Thyroid, German, Titanic [118, 119, 120, 121, 122], Caltech 101 Image Dataset [123] and Corel Dataset [113] are used to compare the classification performances of the proposed approach and state of the art supervised ensemble learning algorithms.

In the FSG, k values of the fuzzy k-NN classifiers are optimized by cross validation using training datasets. In the experiments, fuzzy k-NN is implemented both in Matlab and C++. For C++ implementations, a fuzzified modification of a GPU-based parallel k-NN is used [124]. $T = 10000$ is used as the termination time for Decision Fusion using Decision Margin (DFDM) algorithm. Algorithm parameters of DFDM are selected from the parameter sets

$\lambda \in \{0.01, 0.02, \ldots, 2\}$, $\theta \in \{0.01, 0.02, \ldots, 2\}$, $\bar{\alpha} \in \{0.01, 0.02, \ldots, 2\}$ by cross validation using training datasets.

Classification performances of the proposed algorithms are compared with the state of the art ensemble learning algorithms, such as Adaboost [125], Rotation Forest [126] and Random Subspace [72]. Weighted majority voting is used as the combination rule in Adaboost. Decision trees are implemented as the weak classifiers in both Adaboost and Rotation Forest, and k-NN classifier is implemented as the weak classifier in Random Subspace. The number of weak classifiers $Num_{weak} \in \{1, 2, \ldots, 2D\}$ is selected using cross-validation in the training set, where $D = \sum_{j=1}^{J} D_j$ is the dimension of the feature space of the samples in the datasets.

Experimental analyses of the proposed algorithms on synthetic datasets are given in Section 3.7.1. In Section 3.7.2, classification performances of the proposed algorithms and the state-of-the art classification algorithms are compared using benchmark datasets.

3.7.1 Experiments on Synthetic Datasets

The relationship between the performance of the $k = 1$ and $k \geq 2$ nearest neighbor algorithms and the statistical properties of the datasets has been studied in the last decade by many researchers. Cover and Hart [13] analyzed this relationship with an elegant example, which is revised later by Devroye, Gyorfi and Lugosi [127].

In the example, suppose that the feature vectors of the samples of a training dataset $\{\hat{x}_i\}_{i=1}^{N}$ are grouped in two disks with centers $\bar{o}_1$ and $\bar{o}_2$, which represent the class groups ω_1 and ω_2 such that $\| \bar{o}_1 - \bar{o}_2 \|_2 \geq \sigma_{BC}^{1,2}$ in a two dimensional feature space, where $\sigma_{BC}^{1,2}$ is the between-class variance. In addition, assume that the class conditional densities are uniform and

$$P(\omega_1) = P(\omega_2) = \frac{1}{2}.$$

Note that the probability that n samples belong to the first class ω_1, i.e. that the feature vectors reside in the first disk, is

$$\frac{1}{2^N}\binom{N}{n}.$$

Now, assume that the feature vector of a training sample $\hat{x}_i$ belonging to ω_1 is classified by $k = 1$ nearest neighbor rule. Then, $\hat{x}_i$ will be misclassified if its nearest neighbor resides in the second disk. However, if the nearest neighbor of $\hat{x}_i$ resides in the second disk, then each of the feature vectors must reside in the second disk. Therefore, the classification error is the probability that all of the samples reside in the second disk such that

$$P(y_i \in \omega_1, y_{j \neq i} \in \omega_2) + P(y_i \in \omega_2, y_{j \neq i} \in \omega_1) = \frac{1}{2^N}.$$

If k-NN rule is used for classification with $k = 2\hat{k} + 1$, where $\hat{k} \geq 1$, then an error occurs if $\hat{k}$ or less number of features reside in the first disk with probability

$$P(y_i \in \omega_1, \sum_{j=1}^{N} I(y_{j \neq i} \in \omega_1) \leq \hat{k}) + P(y_i \in \omega_2, \sum_{j=1}^{N} I(y_{j \neq i} \in \omega_2) \leq \hat{k})$$

which is a Binomial distribution $Binomial(\hat{k}, N, \frac{1}{2})$

$$\sum_{n=0}^{\hat{k}} \binom{N}{n} (\frac{1}{2})^n (1-\frac{1}{2})^{N-n} = (\frac{1}{2})^N \sum_{n=0}^{\hat{k}} \binom{N}{n}.$$

Then the following inequality holds

$$(\frac{1}{2})^N \sum_{n=0}^{\hat{k}} \binom{N}{n} > (\frac{1}{2})^N.$$

Therefore, the classification or generalization error of the k-NN depends on the class conditional densities [13] such that $k = 1$ rule performs better than $k \geq 2$ rule when the between class variance of the data distributions $\sigma_{BC}^{c,c'}$ is smaller than the within class variances Σ_c, $\forall c \neq c'$, $c = 1, 2, \ldots, C$, $c' = 1, 2, \ldots, C$.

Although Cover and Hart [13] introduced this example to analyze the classification performances of the nearest neighbor rules, Hastie and Tibshirani [117] used the results of the example in order to develop a metric, which is a function of $\sigma_{BC}^{c,c'}$ and Σ_c, to minimizes the difference between the N-sample and large-sample errors. Since the minimization of error difference is one of the motivations of FSG, a similar experimental setup is designed in order to analyze the performance of FSG by Ozay [94]. In this chapter, the experiments are designed to compare the classification performances of FSG (Algorithm 1), decision fusion with the decision margin (DFDM) given in Algorithm 2 and two sample selection algorithms (Algorithm 3 and Algorithm 4) which are used for spurious training data elimination from meta-layer (SS-1) and base-layer input training datasets (SS-2).

In the experiments, feature vectors of the samples in the datasets are generated using a Gaussian distribution in each $D_j = 2$ dimensional feature space F_j, $j = 1, 2, \ldots, J$. While constructing the datasets, the mean vector $\bar{o}_c$ and the covariance matrix Σ_c of the class-conditional density of a class ω_c

$$f(\bar{x} |\ \bar{o}_c,\ \Sigma_c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left[-\frac{1}{2} (\bar{x} - \bar{o}_c)^T \Sigma_c^{-1} (\bar{x} - \bar{o}_c) \right] \tag{3.35}$$

are systematically varied in order to observe the effect of the class overlaps to the classification performance. One can easily realize that there are explosive alternatives for changing the parameters of the class-conditional densities in a D_j-dimensional vector space. However, it is quite intuitive that the amount of overlaps among the classes affects the performance of the individual classifiers rather than the changes in the class scatter matrix. Therefore, we suffice to control only the amount of overlaps during the experiments. This task is achieved by fixing the covariance matrix Σ_c, in other words within-class-variance, and changing the mean values of the individual classes, which varies the between-class variances, $\sigma_{BC}^{c,c'}$, $\forall c \neq c'$,$c = 1, 2, \ldots, C$, $c' = 1, 2, \ldots, C$.

Denoting v_i as the eigenvector and ϑ_i as the eigenvalue of Σ, we have $\sum v_i = \vartheta_i v_i$. Therefore, the central position of the sample distribution constructed by datasets in a 2-dimensional space is defined by v_1 and v_2 and the propagation is defined by $\vartheta_1^{1/2}$ and $\vartheta_2^{1/2}$. In the datasets, covariance matrices are held fix and equal. Therefore, the eigenvalues represented on both axes are the same. As a result, datasets are generated by the circular Gaussian function with fixed radius.

In this set of experiments, a variety of artificial datasets is generated in such a way that most of the samples are correctly labeled by at least one base-layer classifier. In other words, feature

spaces are generated to construct classifiers which are expert on specific classes. The number of samples belonging to each class ω_c is taken as 250, and 2-dimensional feature spaces are fed to each base-layer classifier as input for $C = 12$ classes with total of 3000 samples. The performances of the classifiers are controlled by fixing the covariance matrices, and changing the mean values of Gaussian distributions which are used to generate the feature vectors.

In order to avoid the misleading information in this gradual overlapping process, the classes are first generated apart from each other to assure the linear separability in the initialization step. Then, the distances between the mean values of the classes are gradually decreased. The ratio of decrease is selected as one tenth of between-class variance of each pair of class ω_c and $\omega_{c'}$, $\forall c \neq c'$, $c = 1, 2, \ldots, C$, $c' = 1, 2, \ldots, C$, which is $\frac{1}{10}\sigma_{BC}^{c,c'}$, where $\sigma_{BC}^{c,c'} = \|\bar{o}_c - \bar{o}_{c'}\|$. The termination condition for the algorithms is

$$\sum_{c,c'} \sigma_{BC}^{c,c'} = 0, \forall c \neq c', c = 1, 2, \ldots, C, c' = 1, 2, \ldots, C.$$

At each epoch, only the mean value of the distribution of one of the classes approaches to the mean value of that of another class, while keeping the rest of the mean values fixed. Defining J as the number of classifiers fed by J different feature extractors and C as the number of classes, the data generation method is given in Algorithm 5.

input : The number of feature spaces J, the number of classes C, the mean value vectors $\bar{o}_c$ and the within class variances Σ_c of the class conditional densities, $\forall c = 1, 2, \ldots, C$.
output: Training and test datasets.
foreach $j = 1, 2, \ldots, J$ **do**
  foreach $c = 1, 2, \ldots, C$ **do**
1 Generate feature vectors using (3.35);
  end
end
foreach $j = 1, 2, \ldots, J$ **do**
  foreach $c' = 1, 2, \ldots, C$ **do**
2 Initialize $\hat{o}_{c'}$;
    foreach $c = 1, 2, \ldots, C$ **do**
      repeat
3 $\sigma_{BC}^{c,c'} \leftarrow \|\bar{o}_c - \hat{o}_{c'}\|$;
4 $\hat{o}_{c'} \leftarrow \bar{o}_c + \frac{1}{10}\sigma_{BC}^{c,c'}$;
      until $\sigma_{BC}^{c,c'} \neq 0$;
    end
  end
end
5 Randomly split the feature vectors into two datasets, namely test and training datasets.

Algorithm 5: The synthetic data generation algorithm.

In the experiments, classification performances of FSG, Decision Fusion using Decision Margin (DFDM) that is given in Algorithm 2 and two sample selection algorithms, that are given Algorithm 3 (SS-1) and Algorithm 4 (SS-2), are compared.

In the first set of the experiments, 7 base-layer classifiers are used. The feature sets are

prepared with fixed and equal $Cov_j = [\Sigma_1 \dots \Sigma_c]^T$, where

$$Cov_j = \begin{pmatrix} 5 & 5 \\ 5 & 5 \end{pmatrix}$$

is the covariance matrix of the class conditional distributions in F_j, $\forall c, c' = 1, 2, \dots, 12$, $\forall j = 1, 2, \dots, 7$. In other words, $\vartheta_1^{\frac{1}{2}} = 5$ and $\vartheta_2^{\frac{1}{2}} = 5$.

The features are distributed with different $\sigma_{BC}^{c,c'}$ and converged towards each other using Algorithm 5. The matrix $\Omega_j = [\bar{o}_{c,j}]_{c=1}^{12}$, with the row vectors that contain the mean values $\bar{o}_{c,j}$ of the distribution of each class ω_c at each space $j = 1, 2, \dots, 7$ is defined as

$$\Omega = [\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5, \Omega_6, \Omega_7].$$

In addition, this experimental apparatus enables us to analyze the relationship between the number of samples which are correctly classified by at least one of the base-layer classifiers and classification performances of the proposed algorithms. Therefore, the average number of samples which are correctly classified by at least one base-layer classifier, which is denoted as $\hat{Ave}_{corr}$, is also given in the experimental results.

In each epoch, features belonging to different classes are distributed with different topologies in each classifier by different overlapping ratios. For example, feature vectors of the samples belonging to the ninth class is located apart from that of the rest of the classes in F_7, while they are overlapped in other feature spaces. In this way, the classification behaviors of the base-layer classifiers are controlled through the topological distributions of the features, and classification performances are measured by the metrics given in the previous Section 3.3.

In Table 3.1, performances of individual classifiers and the proposed algorithms are given for an instance of the dataset generated by Algorithm 5, where the datasets are constructed in such a way that each sample is correctly recognized by at least one of the base-layer classifiers, i.e. $\hat{Ave}_{corr} = 1$. Although the performances of individual classifiers are in between $53\% - 66\%$, the classification performance of FSG is 99.9%. Performance boost is obtained by the employment of the decision weighting and sample selection algorithms. In that case, different classes are distributed at higher relative distances and with different overlapping ratios. The matrix Ω used in the first experiment is

$$\begin{bmatrix}
-10 & -10 & -10 & -10 & -10 & -10 & -10 & -10 & -10 & -10 & 10 & -15 & -25 & -25 \\
-10 & 10 & -10 & 10 & -10 & 10 & -10 & 10 & -25 & -25 & 0 & 0 & -15 & 10 \\
10 & -10 & 10 & -10 & 10 & -10 & 20 & -10 & 15 & -15 & -10 & -10 & -25 & -25 \\
15 & 15 & 15 & 15 & 25 & 25 & 15 & 15 & 15 & 15 & 10 & 10 & -15 & 10 \\
15 & 5 & -25 & 0 & -15 & 5 & -15 & 5 & -15 & 5 & 15 & 15 & 5 & -10 \\
-25 & 0 & 15 & 5 & 15 & 5 & 15 & 5 & 15 & 5 & 15 & 5 & 0 & 0 \\
5 & 15 & 5 & 15 & 5 & 15 & 5 & 15 & 5 & 15 & 10 & 15 & -25 & 25 \\
5 & -20 & 5 & -20 & 5 & -20 & 5 & -15 & 5 & -15 & -15 & -10 & 25 & -25 \\
-5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & 15 & 10 & 25 & 25 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 0 & 0 & 25 & 0 \\
-5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -15 & 10 & -10 & 10 \\
5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 25 & -25 & 10 & -10
\end{bmatrix}.$$

Table 3.1: Comparison of the performances (perf. %) of the base-layer classifiers with respect to the classes (C) and the performances of the FSG, Decision Fusion using Decision Margin (DFDM) and two sample selection algorithms SS-1 and SS-2, when $\hat{Ave}_{corr} = 1$.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	FSG	DFDM	SS-1	SS-2
C1	66.0	63.6	67.6	62.8	61.6	85.6	50.0	**100**	**100**	**100**	**100**
C2	67.2	60.8	49.6	50.8	***98.4***	38.4	36.8	**100**	**100**	**100**	**100**
C3	54.4	58.8	50.8	***85.2***	72.4	53.6	47.6	99.2	**100**	**100**	**100**
C4	66.8	64.0	***96.8***	66.4	61.6	22.8	37.6	**100**	**100**	**100**	**100**
C5	60.8	***90.0***	56.0	63.6	75.2	38.8	48.4	**100**	**100**	**100**	**100**
C6	***91.6***	57.2	69.6	54.0	66.0	43.6	73.6	**100**	**100**	**100**	**100**
C7	57.2	55.2	65.2	57.6	60.8	37.2	94.4	**100**	**100**	**100**	**100**
C8	78.4	75.6	86.0	69.2	54.4	61.6	***97.6***	**100**	**100**	**100**	**100**
C9	40.8	41.2	36.0	36.0	32.8	26.0	***99.6***	**100**	**100**	**100**	**100**
C10	44.0	32.4	32.0	38.0	37.6	43.2	***95.6***	**100**	**100**	**100**	**100**
C11	32.0	35.2	33.6	40.0	39.6	***92.8***	38.8	99.6	**100**	**100**	**100**
C12	37.6	39.6	34.4	52.0	44.4	***97.2***	63.6	99.6	**100**	**100**	**100**
AP	58.0	56.1	56.5	56.3	58.7	53.4	65.3	99.9	**100**	**100**	**100**

In Table 3.2, the performance results of algorithms at another epoch of the experiments are given. In this experiment, 90% of the samples are correctly classified by at least one of the base-layer classifiers, i.e. $\hat{Ave}_{corr} = 0.9$. The corresponding mean value matrix Ω of each class at each feature space is

$$\begin{bmatrix}
-20 & -20 & -10 & -10 & -10 & -10 & -10 & -10 & 10 & -10 & 10 & -15 & 15 & 5 \\
-20 & 20 & -10 & 10 & -10 & 10 & -10 & 10 & -5 & -10 & 0 & 0 & -5 & 10 \\
10 & -10 & 20 & -20 & 10 & -10 & 10 & -10 & 15 & -15 & -10 & -10 & -10 & -5 \\
15 & 15 & 25 & 25 & 5 & 5 & -5 & 10 & 15 & 15 & 10 & 10 & -15 & 10 \\
15 & 5 & -5 & 0 & -25 & 25 & -10 & 5 & -5 & 5 & 15 & 15 & 5 & -10 \\
-5 & 0 & 15 & 5 & 25 & 25 & 15 & 5 & 15 & 5 & 15 & 5 & 0 & 0 \\
5 & 15 & 5 & 15 & 5 & 15 & 25 & 25 & 5 & 10 & 10 & 15 & -5 & 5 \\
5 & -20 & 5 & -10 & 5 & -5 & 25 & 25 & 5 & -15 & -15 & -10 & 5 & -5 \\
-5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & -25 & -25 & 15 & 10 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 25 & 25 & 0 & 0 & 5 & 0 \\
-5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -25 & 25 & -10 & 10 \\
5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 15 & -15 & 25 & -25
\end{bmatrix}.$$

Table 3.2: Comparison of the performances (perf. %) of individual classifiers with respect to the classes (C) and the performances of the FSG, Decision Fusion using Decision Margin (DFDM) and two sample selection algorithms SS-1 and SS-2, when $\hat{Ave}_{corr} = 0.9$.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	FSG	DFDM	SS-1	SS-2
C1	***97.2***	67.6	68.4	69.6	28.0	53.6	65.6	**100**	**100**	**100**	**100**
C2	***96.8***	63.2	63.6	41.6	67.6	44.4	30.0	**100**	**100**	**100**	**100**
C3	56.4	***95.2***	57.2	66.8	56.8	47.2	66.4	99.6	**100**	99.9	99.9
C4	60.8	***98.0***	22.8	30.8	62.0	24.4	46.0	**100**	**100**	**100**	**100**
C5	56.8	24.0	***96.8***	27.2	44.8	38.8	50.4	**100**	**100**	**100**	**100**
C6	32.8	68.4	***97.6***	71.2	57.2	43.6	14.0	**100**	**100**	**100**	**100**
C7	54.0	65.6	74.4	***96.8***	52.4	36.8	24.4	99.6	**100**	**100**	**100**
C8	77.2	43.6	29.6	***98.4***	48.0	65.6	27.6	99.6	**100**	99.9	99.9
C9	45.2	34.0	35.2	35.2	***98.8***	24.8	29.2	**100**	**100**	**100**	**100**
C10	40.0	33.6	22.4	47.6	***90.4***	33.6	18.0	**100**	**100**	**100**	**100**
C11	49.2	28.4	38.0	28.0	38.4	***100***	26.0	**100**	**100**	**100**	**100**
C12	34.8	34.4	22.4	34.4	44.4	65.2	***98.8***	**100**	**100**	**100**	**100**
AP	58.4	54.6	52.3	53.9	57.4	48.1	41.3	99.9	**100**	99.98	99.98

In the third set of the experiments, samples are distributed in the descriptors such that 80% of the samples are correctly classified by at least one base-layer classifier ($\hat{Ave}_{corr} = 0.8$). The performance results of the experiment are provided in Table 3.3 and the corresponding mean value matrix Ω is

$$\begin{bmatrix}
-12 & -12 & -7.5 & -7.5 & -10 & -10 & -7.5 & -7.5 & 10 & -10 & 10 & -15 & 10 & 5 \\
-10 & 10 & -8 & 8 & -10 & 10 & -10 & 10 & -5 & -10 & 0 & 0 & -5 & 10 \\
10 & -10 & 10 & -15 & 10 & -10 & 10 & -10 & 10 & -15 & -10 & -10 & -5 & -5 \\
15 & 15 & 15 & 17.5 & 5 & 5 & -5 & 10 & 15 & 15 & 10 & 10 & -15 & 10 \\
15 & 5 & -5 & 0 & -15 & 15 & -10 & 5 & -5 & 5 & 15 & 15 & 5 & -10 \\
-5 & 0 & 15 & 5 & 15 & 15 & 10 & 5 & 10 & 5 & 10 & 5 & 0 & 0 \\
5 & 15 & 5 & 15 & 5 & 15 & 10 & 15 & 5 & 10 & 10 & 15 & -5 & 5 \\
5 & -15 & 5 & -10 & 5 & -5 & 15 & -15 & 5 & -15 & -15 & -10 & 5 & -5 \\
-5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & -10 & -15 & 15 & 10 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 20 & 20 & 0 & 0 & 5 & 0 \\
-5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 10 & -10 & 10 \\
5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 15 & -15 & 10 & -15
\end{bmatrix}.$$

Table 3.3: Comparison of the performances (perf. %) of individual classifiers with respect to the classes (C) and the performances of the FSG, Decision Fusion using Decision Margin (DFDM) and two sample selection algorithms SS-1 and SS-2, when $\hat{Ave}_{corr} = 0.8$.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	FSG	DFDM	SS-1	SS-2
C1	***82.8***	63.6	66.0	71.2	32.0	54.0	67.2	99.6	**100**	**100**	**100**
C2	***73.2***	63.6	48.0	34.4	51.6	37.6	29.6	97.2	**100**	99.5	98.7
C3	55.2	***78.0***	59.6	51.2	62.4	46.8	69.6	98.4	**100**	99.1	99.0
C4	61.2	***82.0***	26.0	31.2	44.4	17.6	52.8	98.4	**100**	98.8	99.2
C5	53.2	23.2	***76.8***	29.6	41.2	39.6	45.2	**100**	**100**	**100**	**100**
C6	24.8	66.4	***87.2***	62.0	56.4	42.4	21.2	98.8	**100**	99.2	99.0
C7	54.0	63.2	54.8	***88.4***	55.2	36.8	23.6	98.4	**100**	99.0	99.2
C8	***80.8***	39.2	22.8	74.8	45.2	63.2	23.6	96.4	**99.2**	96.6	97.8
C9	39.6	33.2	33.2	29.6	***83.6***	21.6	29.6	99.2	**100**	**100**	**100**
C10	38.4	35.6	30.8	47.6	***82.8***	38.0	24.0	99.2	**100**	**100**	**100**
C11	33.2	30.0	30.8	30.4	38.8	***84.4***	29.6	96.4	**100**	96.8	97.2
C12	40.4	33.2	28.0	40.4	32.4	58.8	***81.2***	99.2	**100**	**100**	**100**
AP	53.1	50.9	47.0	49.2	52.2	45.1	41.4	98.4	**99.9**	99.1	98.8

In the fourth set of the experiments given in Table 3.4, samples are distributed in the descriptors such that each classifier can correctly classify 70% of the samples. The corresponding mean value matrix Ω is

$$\begin{bmatrix} -12.5 & -12.5 & -10 & -10 & -10 & -10 & -10 & -10 & 10 & -10 & 10 & -15 & 15 & 5 \\ -10 & 15 & -10 & 10 & -10 & 10 & -10 & 10 & -5 & -10 & 0 & 0 & -5 & 10 \\ 10 & -10 & 15 & -15 & 10 & -10 & 10 & -10 & 15 & -15 & -10 & -10 & 10 & -5 \\ 15 & 15 & 19 & 19 & 5 & 5 & -5 & 10 & 15 & 15 & 10 & 10 & -15 & 10 \\ 15 & 5 & -5 & 0 & -17.5 & 17.5 & -10 & 5 & -5 & 5 & 15 & 15 & 5 & -10 \\ -5 & 0 & 15 & 5 & 17.5 & 17.5 & 15 & 5 & 15 & 5 & 15 & 5 & 0 & 0 \\ 5 & 15 & 5 & 15 & 5 & 15 & 17.5 & 17.5 & 5 & 10 & 10 & 15 & -5 & 5 \\ 5 & -20 & 5 & -10 & 5 & -5 & 17.5 & -17.5 & 5 & -15 & -15 & -10 & 5 & -5 \\ -5 & -5 & -5 & -5 & -5 & -5 & -5 & -5 & -15 & -15 & 15 & 10 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 22.5 & 22.5 & 0 & 0 & 5 & 0 \\ -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -10 & 10 & -10 & 10 \\ 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 5 & -5 & 15 & -15 & 15 & -15 \end{bmatrix}. \quad (3.36)$$

Note that, the performance of the overall FSG decreases as the percentage of the samples that are correctly classified by at least one classifier decreases, i.e. $\hat{Ave}_{corr}$ decreases. This observation is due to the results given in the previous section which state that the large-sample classification error of the meta-layer classifier of the FSG is bounded by Bayes Error, which can be achieved if each sample is correctly classified by at least one base-layer classifier such that the features of samples belonging to the same class reside in the same Voronoi regions in the fusion space.

In order to satisfy this requirement, SS-1 and SS-2 algorithms have been employed in the experiments, and they boost the performance of the FSG. In addition, the amount of their performance boost increases as $\hat{Ave}_{corr}$ decreases. Among all the proposed algorithms, the best average performance is obtained by DFDM. This is due to the fact that the DFDM

Table 3.4: Comparison of the performances (perf. %) of individual classifiers with respect to the classes (C) and the performances of the FSG, Decision Fusion using Decision Margin (DFDM) and two sample selection algorithms SS-1 and SS-2, when $\hat{Ave}_{corr} = 0.7$.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	FSG	DFDM	SS-1	SS-2
C1	***75***	42	68	52	36	62	46	99	**100**	**100**	**100**
C2	***64***	45	41	38	43	37	32	98	**100**	99	99
C3	46	***72***	60	40	39	52	46	88	**94**	90	91
C4	68	***72***	23	33	45	17	59	98	**100**	**100**	**100**
C5	54	22	***70***	28	40	42	32	**100**	**100**	**100**	**100**
C6	22	68	***74***	50	46	28	18	97	**100**	99	99
C7	65	62	50	***72***	44	34	20	96	**99**	**99**	**99**
C8	55	30	25	***75***	44	61	18	89	**95**	94	**95**
C9	36	24	36	30	***67***	32	23	**100**	**100**	**100**	**100**
C10	42	32	24	27	***74***	32	21	98	**100**	**100**	**100**
C11	31	17	34	16	38	***70***	26	95	**99**	97	97
C12	33	28	27	41	38	67	***68***	**100**	**100**	**100**	**100**
AP	49.3	42.9	44.3	41.8	46.1	44.4	34.2	96.4	**98.9**	98.2	98.3

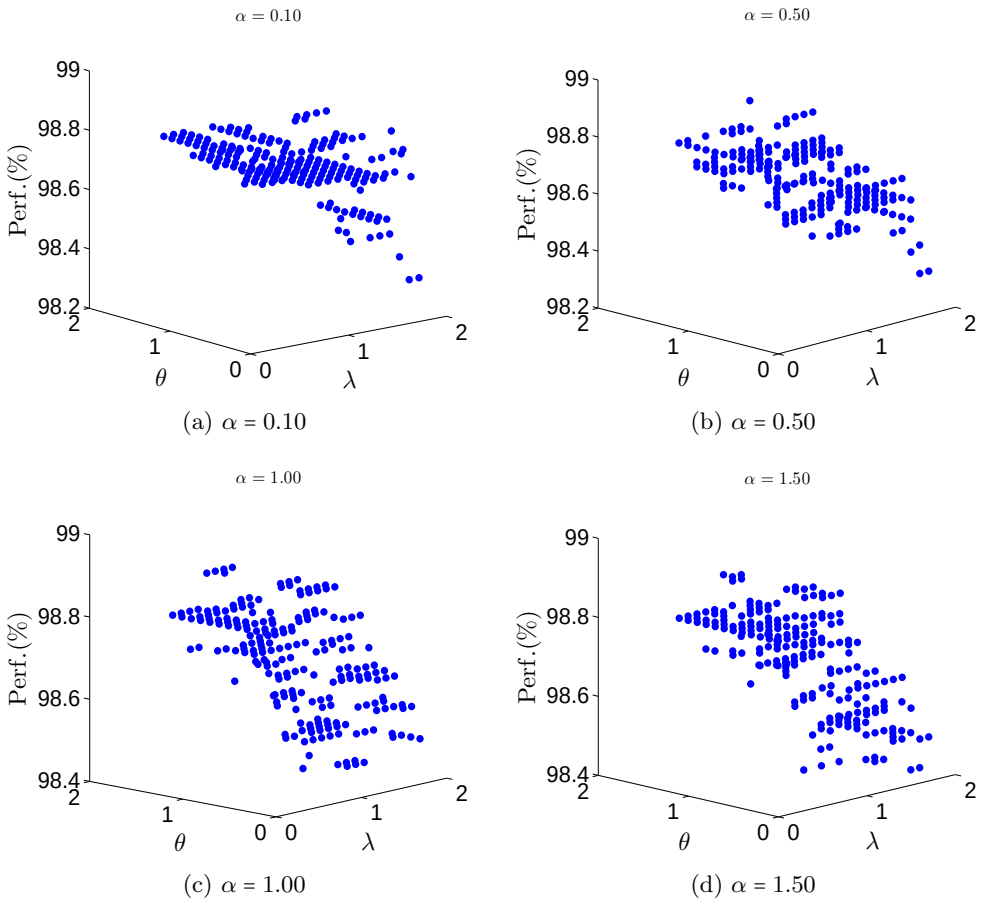
algorithm converges to the optimal solution if the Lagrangian of the problem computed on the data has a saddle point, i.e. the solution exits. When the optimal solution is obtained, decision vectors of base-layer classifiers or meta-layer feature vectors of samples are aggregated by minimum decision margin, i.e. the minimum classification error is achieved at the meta-layer. However, the number of samples eliminated by SS-1 and SS-2 algorithms increases as $\hat{Ave}_{corr}$ decreases. Therefore, sample elimination methods may damage the requirements of the existence of points of non-zero probability measures or continuity points of the densities in the datasets [13]. If the eliminated samples cause singularities which violate these two requirements, then the performance boost is not assured.

3.7.1.1 Convergence Analysis of Decision Fusion using Decision Margin Algorithm

Finally, the relationship between the classification performance of Decision Fusion using Decision Margin Algorithm (DFDM) and its parameters is depicted in Figure 3.4 for the experimental results given in Table 3.4.

The results show that the classification performance is more sensitive to λ and θ than the initial value of α which is used to initialize a vector of scaled dual optimization variables $\bar{u}$. Because, $\bar{u}$ is updated in the iterations of the algorithm and is converged to an optimal solution if there exits an optimal solution that can be achieved using the training dataset and there is no duality gap [112].

In this experiment, the selection of the algorithm parameters affected the classification performance by %0.6 since the difference between the maximum and the minimum performance obtained by DFDM algorithm is 98.9% – 98.3% = 0.6%. Therefore, a *reasonable* convergence error for the computation of decision weights can be obtained using the suggested optimization method in the minimization of decision margins in the DFDM algorithm.



(a) $\alpha = 0.10$ (b) $\alpha = 0.50$

(c) $\alpha = 1.00$ (d) $\alpha = 1.50$

Figure 3.4: The relationship between the classification performance (Perf. (%)) of the suggested DM algorithm and its parameters.

3.7.2 Experiments on Benchmark Datasets

In the experiments, classification performances of $k = 1$ nearest neighbor rule, Fuzzy Stacked Generalization (FSG), Decision Fusion using Decision Margin (DFDM), two sample selection algorithms SS-1 and SS-2, and the state of the art algorithms, Adaboost, Random Subspace (RS) and Rotation Forest (RF), are compared using benchmark datasets.

Experiments on the benchmark datasets are performed in two groups, namely, the experiments on i) multi-attribute and ii) multi-feature datasets. In the multi-attribute experiments, feature vectors consisting of multiple attributes reside in a single feature space $F_j = F_{1j} \times \ldots \times F_{aj} \times \ldots \times F_{Aj}$. In these experiments, decision fusion algorithms are implemented by employing individual base-layer classifiers on a feature space F_{aj} consisting of an each individual attribute. In multi-feature experiments, each base-layer classifier is employed on an individual feature space F_j, $\forall j = 1, 2, \ldots, J$. State of the art algorithms have been employed on an aggregated feature space $F = F_1 \times \ldots \times F_j \times \ldots \times F_J$ in multi-attribute and multi-feature experiments.

In the next subsection, state of the art ensemble learning algorithms which have been used in the experiments for performance comparison are briefly explained.

3.7.2.1 A Brief Description of State of the Art Ensemble Learning Algorithms

In this subsection, we provide a short overview for the ensemble learning algorithms employed in this study, for comparing the suggested supervised decision fusion methods. The algorithms, namely, Adaboost, Rotation Forest and Random Subspaces are the most popular ensemble learning algorithms available in the literature. It is well-known that many variants of these algorithms are available in the literature. In this study, we employ the following basic versions of these algorithms.

3.7.2.1.1 Adaboost

In the following, the Adaboost algorithm is explained as an ensemble learning approach for decision fusion, in which a collection of *weak* base-layer classifiers are generated and combined using a combination rule to construct a *stronger* meta-layer classifier which performs better than base-layer classifiers [125].

Given a set of training samples $S = \{(s_i, y_i)\}_{i=1}^{N}$, at each iteration $t = 1, 2, \ldots, T$ of the algorithm, a decision or hypothesis $f_t(\cdot)$ of the weak classifier is computed with respect to the distribution $p_t(\cdot)$ on the training samples at t by minimizing the weighted error $\epsilon_t = \sum_{i=1}^{N} p_t(i) I(f_t(s_i) \neq y_i)$, where $I(\cdot)$ is the indicator function. The distribution is initialized uniformly $p_1(i) = \frac{1}{N}$ at $t = 1$, and is updated by $\frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$ as follows

$$p_{t+1}(i) = \frac{p_t(i) e^{-\frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t}) y_i f_t(s_i)}}{Z_t}, \tag{3.37}$$

where Z_t is a normalization parameter, called *partition function*. At the output of the algorithm, a strong classifier $H(\cdot)$ is constructed for a sample s' using

$$H(s') = \mathsf{sign}\left(\sum_{t=1}^{T} \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) f_t(s')\right).$$

3.7.2.1.2 Random Subspace

The decision fusion strategy of Random Subspace [72] is similar to that of Adaboost and FSG. Briefly, J feature subspaces F_j, $j = 1, 2, \ldots, J$ are selected from a set of feature spaces F, *randomly* and *without replacement*. Then, each base-layer classifier is trained on each randomly selected F_j to predict the base-layer decision, such as a class membership value $\mu_c(\bar{x}_{i,j})$. At the meta-layer, base-layer decisions are fused by majority voting or averaging as

$$\mu_c(\bar{x}_i) = \frac{1}{J} \sum_{j=1}^{J} \mu_c(\bar{x}_{i,j}), \forall i, c.$$

3.7.2.1.3 Rotation Forest

Rotation Forest employs a *forest* (an ensemble) of decision trees as base-layer classifiers. In each base-layer classifier, randomly selected feature vectors are rotated according to their statistical correlation by preserving their order in feature spaces using Principle Component

Analysis (PCA) algorithm [126]. Then, each classifier predicts the class memberships of samples using the rotated features. Finally, a meta-layer classifier linearly combines decisions of the base-layer classifiers to compute the final decision. Formally, the algorithm can be summarized in the following steps;

1. For each base-layer classifier f_a, $a = 1, 2, \ldots, A$, randomly split a set of features into J subsets $F_{a,j}$, $j = 1, 2, \ldots, J$,

   (a) Eliminate from a dataset $S_{a,j} \subset S$, which contains the samples that are represented by the features in $F_{a,j}$, a random subset of classes,

   (b) Draw a bootstrap sample $S'_{a,j}$ from $S_{a,j}$,

   (c) Apply PCA on $S'_{a,j}$ to obtain a matrix of $\mathfrak{C}_{a,j}$ of the coefficients of principle components,

   (d) Construct a block diagonal rotation matrix $\mathfrak{R}_a$ by arranging the matrices $\mathfrak{C}_{a,j}$ in the order of subset indices,

   (e) Then, rearrange the columns of $\mathfrak{R}_a$ by matching the order of features in F in a new matrix $\mathfrak{R}^o_a$,

   (f) In order to update feature vectors for the base-layer classifier, project the original features in F with $\mathfrak{R}^o_a$ to a new feature space F'_a.

   (g) Finally, train the base-layer classifier with F'_a to predict the class membership values of the samples $\mu_c(\bar{x}_{i,a}\mathfrak{R}^o_a)$ as the classifier decision, where $\bar{x}_{i,a}\mathfrak{R}^o_a$ is the product of $\mathfrak{R}^o_a$ by $\bar{x}_{i,a}$, $\forall i = 1, 2, \ldots, N$ and $\forall c = 1, 2, \ldots, C$.

2. At the meta-layer, compute the meta-layer decision by the average combination of the decisions of base-layer classifiers as

$$\mu_c(\bar{x}_i) = \frac{1}{A} \sum_{a=1}^{A} \mu_c(\bar{x}_{i,a}\mathfrak{R}^o_a), \forall i, c.$$

3.7.2.2 Experiments on Multi-attribute Datasets

In the experiments, two-class Breast Cancer (BCancer), Diabetis, Flare Solar (FSolar), Thyroid, German, Titanic [118, 119, 120, 121, 122] datasets are used as multi-attribute datasets. The number of attributes of the feature vectors of the samples in the datasets are given in Table 3.5. Training and test datasets are randomly selected from the datasets. The experiments are repeated 100 times and the average performance values are given in the table. The ratio of the number of samples belonging to MC to the size of the sample dataset is denoted as PMC. Performances of the benchmark algorithms and the proposed decision fusion methods are also given in Table 3.5.

An interesting observation from Table 3.5 is that the $k = 1$ nearest neighbor rule outperforms various well-known ensemble learning algorithms such as Adaboost and Rotation Forest, if the number of attributes is small, e.g. the dimension $D = 3$.

Experimental results show that the suggested DFDM algorithm performs better than all of the aforementioned algorithms in all the tests. In addition, the performance gains of the

sample selection and decision weighting algorithms increase with increasing PMC. In addition, SS-2 algorithm outperforms SS-1 algorithm in the experiments where the removal of the samples from the training dataset damages the statistical stability between training and test datasets. In this case, re-training the base-layer classifiers may recover the statistical stability [84]. However, SS-2 may perform worse than SS-1 in the experiments where re-training the base-layer classifiers may not successfully recover the stability, such as in Thyroid dataset. Moreover, re-training the classifiers based on $\hat{S}$ may further increase the divergence between the distribution of features in S_{meta} and S'_{meta}.

Table 3.5: Classification performances of the algorithms on Multi-attribute Datasets.

Datasets	**FSolar**	**German**	**Titanic**	**BCancer**	**Diabetis**	**Thyroid**
Num. of Att.	9	20	3	9	8	5
Adaboost	66.21%	75.89%	75.06%	74.87%	75.98%	93.10%
RF	62.75%	74.81%	70.14%	70.58%	72.43%	95.64%
RS	65.04%	75.17%	74.83%	74.08%	74.40%	94.78%
1 NN	60.58%	71.12%	75.54%	67.30%	69.88%	95.64%
FSG	67.33%	75.30%	76.01%	75.51%	77.42%	96.41%
SS-1	68.27%	77.35%	78.14%	76.42%	77.55%	96..62%
SS-2	69.10%	78.12%	78.69%	77.13%	78.14%	96.60%
DFDM	72.49%	80.13%	81.02%	80.15%	79.03%	98.15%
PMC (%)	7.65%	7.21%	13.13%	8.18%	0.83%	0.06%

3.7.2.3 Experiments on Multi-feature Datasets

In this section, the algorithms have been analyzed on two image classification benchmark datasets, which are Corel Dataset consisting of 600 classes and Caltech 101 Dataset consisting of 102 classes.

3.7.2.3.1 Experiments on Corel Dataset

In the Corel Dataset experiments, 4 to 8 feature (descriptor) combinations of the MPEG-7 features are used over 10 to 30 classes, each of which contains 100 samples from the dataset. 50 of the samples of each class are used for the training, and the remaining 50 samples are used for testing.

The employed Haar and 7 of MPEG-7 visual features (descriptors) [128, 129] are summarized as follows

- **Color Structure** (Feature Space Dimension $D_j = 32$) gives information on the spatial statistics and structural properties of colors in the image. A color structure feature is a vector of eight bit-quantized histogram values $\overline{hist}_m(q)$, where m is the scale of a square structuring element and $q \in \{1, 2, \ldots, Q\}$, where $Q \in \{32, 64, 128, 256\}$. For instance, a feature extracted using $m = 8$ gives information about the number of times a color is present in an 8×8 windowed neighborhood which is traversed over the image. Moreover,

if Q = 32, then each variable of the vector represents the relative spatial frequency of each quantized color value in the image.

- **Color Layout** (Feature Space Dimension D_j = 12) captures the spatial arrangement of colors in an image, similar to Color Structure, but the features are invariant to scale unlike Color Structure. In order to achieve scale invariance, an image is partitioned into 8×8 = 64 blocks. Then, a representative color is selected from each block by using a pooling method such as average pooling which computes the average of the pixel colors in a block. Three pooled images are constructed for luminance, the blue and red chrominance values. Next, a Discrete Cosine Transform (DCT) is employed on the pooled images and three sets of DCT coefficients are constructed. In order to group the low frequency coefficients of the 64 blocks, a zigzag scanning is performed with these three sets of DCT coefficients. Therefore, 3 zigzag scanned matrices is obtained for each pooled images. In the experiments, a total of 12 coefficients, 6 for luminance and 3 for each chrominance, are used.

- **Edge Histogram** (Feature Space Dimension D_j = 80) represents the spatial distribution of five types of edges, namely four directional edges (horizontal, vertical, left and right diagonal edges) and a non-directional edge for 16 local regions in the image. Since there are five types of edges for each local region, $16 * 5 = 80$ histogram bins are computed.

- **Region-based Shape** (Feature Space Dimension D_j = 35) considers all the pixels which represent a shape, such as the boundary and interior pixels. Therefore, shapes of the objects consisting of a single connected region or multiple regions with holes can be represented with that feature. First, a shape is decomposed into a number of orthogonal complex-valued 2-D basis functions using Angular Radial Transform which is a 2D complex transform defined on a unit disk in polar coordinates. Then, the normalized and quantized magnitudes of coefficients are used to describe the shape. In the implementations, twelve angular and three radial functions are used to extract 35 features, which are concatenated into a feature vector of Region-based Shape descriptor.

- **Dominant Color** (Feature Space Dimension D_j = 16) provides a compact representation of descriptive colors in an image region or a whole image. A dominant color feature vector is

$$\big((\overline{comp}_1, p_1, \sigma_1), (\overline{comp}_2, p_2, \sigma_2) \ldots, (\overline{comp}_N, p_N, \sigma_N), homog\big),$$

  where N is the number of dominant colors in the image, $\overline{comp}_i$ is a vector of color space component values, p_i is the fraction of pixels in the image or image region corresponding to the i^{th} color such that $\sum_{i=1}^{N} p_i = 1$, σ_i describes the variation of the color values of the pixels in a cluster around the i^{th} color, and *homog* is a single number that represents the overall spatial homogeneity of the dominant colors in the image. In the experiments, 3 dominant colors are computed in RGB space.

- **Scalable Color** (D_j = 64) computes a color histogram of an image in the $\mathcal{HSI}$ color space and applies a Haar transform-based encoding scheme across values of the histogram. In the algorithm, first the histogram values are computed, normalized and nonlinearly mapped into a four-bit integer representation, giving higher significance to small values. Then, the Haar transform is applied to the four-bit integer values across the histogram bins. In the experiments, 64 coefficients of the Haar transform are used, which are equivalent to histograms with 64 bins.

- **Homogenous Texture** (D_j = 62) describes the region texture in an image using the mean energy and the energy deviation from a set of frequency channels which are computed using Gabor filters computed on the image. Gabor features are computed for each angular direction $30° \times a$, where $a \in \{0, 1, 2, 3, 4, 5\}$ is the angular index and radial direction $B_b = B_0 \cdot 2^{-b}$, where $b \in \{0, 1, 2, 3, 4\}$ and B_0 is the largest bandwidth specified by $\frac{1}{2}$. Therefore, 30 frequency channels are computed using Gabor filters. For each i^{th} channel, the energy en_i and the energy deviation dev_i, which are defined as as the log-scaled sum and standard devitation of the square of the Gabor-filtered Fourier transform coefficients of an image, are computed. Concatenating en_i and dev_i, $\forall i = 1, 2, \ldots, 30$, with the mean and the standard deviation of the image, a 62 dimensional vector is constructed as a Homogenous Texture feature vector.

- **Haar Descriptor** (D_j = 195) employs Haar wavelets to hierarchically decompose an image in order to obtain information about the coarseness of the image at different scales. At each scale, three different orientations of Haar wavelets, each of which responds to variances in intensities across different axes, are used. Therefore, the information about how intensity varies in each color channel in the horizontal, vertical and diagonal directions is obtained. In the experiments, each image is scaled into a 128×128 pixel and 4-layer two dimensional Haar wavelet transform is applied. Concatenation of the wavelet coefficients computed at each layer constructs the feature vector,

In the experiments, the following 4 to 8 feature combinations are used to test the behavior of the suggested algorithms to feature space dimensions:

- 4 Features (4FS): Color Structure, Color Layout, Edge Histogram, Region-based Shape,
- 5 Features (5FS): Color Structure, Color Layout, Edge Histogram, Region-based Shape, Haar,
- 6 Features (6FS): Color Structure, Color Layout, Edge Histogram, Region-based Shape, Haar, Dominant Color,
- 7 Features (7FS): Color Structure, Color Layout, Edge Histogram, Region-based Shape, Haar, Dominant Color, Scalable Color, and
- 8 Features (8FS): Color Structure, Color Layout, Edge Histogram, Region-based Shape, Haar, Dominant Color, Scalable Color, Homogenous Texture.

The selected MPEG-7 features have high variance and a well-balanced cluster structure [128]. These properties allow us to distinguish the samples in different classes. In addition, the feature vectors in the descriptors satisfy i.i.d. (independent and identically distribution) conditions by providing high between class variance values [128]. Therefore, the statistical properties of the feature spaces provide wealthy information variability.

In the Corel Dataset, two types of experiments are employed. In the first type of the experiments, samples belonging to a pre-defined set of classes is selected to construct smaller datasets. Then the change of the performance of the algorithms is analyzed as the new samples belonging to the new classes are added to the datasets and the new features are added to the feature sets. In the second type of the experiments, the datasets are constructed by selecting samples belonging to randomly selected classes. In these experiments, the random

class selection procedure is repeated 10 times and the average performance of the experiments for each procedure is given.

The pre-defined class names of 10, 15 and 20 class classification experiments are the following

- **10 Class Classification**: New Guinea, Beach, Rome, Bus, Dinosaurs, Elephant, Roses, Horses, Mountain, and Dining,
- **15 Class Classification**: Classes used in 10 Class Classification together with Autumn, Bhutan, California Sea, Canada Sea and Canada West,
- **20 Class Classification**: Classes used in 15 Class Classification together with China, Croatia, Death Valley, Dogs and England.

When the sample set is fixed, the change of the classification performance is analyzed as the new features are added from combinations of 4FS to 8FS feature sets. The classification results are given in Table 3.6. In the experiments, the Fuzzy Stacked Generalization and its sample selection and decision weighting algorithms outperform the benchmark algorithms. Moreover, the suggested DFDM algorithm gives the highest classification performance among all the classification algorithms.

As new features are added, the performances of the algorithms which employ majority voting to the classifier decision may decrease. For instance, when Dominant Color and Scalable Color features are added to the combination of features in 5FS to construct the 6FS and the 7FS, the classification performances of the FSG and the Random Subspace, which employ majority voting at the meta-layer classifiers, decrease.

However, the algorithms which employ weighted decision fusion at the meta-layer, such as Adaboost, Rotation Forest and DFDM, are more robust to the integration of *spurious* features to the existing feature sets as observed in Table 3.6. This fact is observed specifically in the weighted combination or selection of the decisions of the base-layer classifiers employed on Dominant Color and Scalable Color features; if these features do not provide discriminative information about the samples, or provide destructive information, then lower weights are assigned to the decisions of the classifiers employed on these features.

Note that the PMC decreases when Dominant Color and Scalable Color features are added to the feature set 5FS. Although these features do not provide discriminative information to the base-layer classifiers in order to increase their average classification performances, the classification performances of the sample selection algorithms SS-1 and SS-2 increase in the 6FS and the 7FS. Therefore, Dominant Color and Scalable Color features provide discriminative information about the *specific* samples in MC which are correctly classified by the base-layer classifiers employed on these features.

Table 3.6: Classification results of the benchmark and the proposed decision fusion algorithms on the Corel Dataset with varying number of features and classes.

	Algorithms	4FS	5FS	6FS	7FS	8FS
10-Class Experiments	**Adaboost**	63.0%	63.6%	63.2%	66.6%	67.2%
	Rotation Forest	76.2%	74.4%	74.6%	76.6%	78.2%
	Random Subspace	78.1%	77.5%	75.8%	76.9%	75.5%
	FSG	85.6%	86.8%	85.6%	85.8%	85.8%
	SS-1	86.9%	87.6%	87.4%	87.6%	88.1%
	SS-2	86.4%	87.4%	87.4%	87.6%	88.9%
	DFDM	87.1%	89.9%	89.9%	90.1%	91.2%
	PMC (%)	6.6%	5.1%	5.0%	4.7%	3.6%
15-Class Experiments	**Adaboost**	42.2%	45.5%	43.2%	46.8%	46.8%
	Rotation Forest	60.2%	60.6%	60.9%	60.9%	61.3%
	Random Subspace	65.5%	64.1%	59.8%	63.3%	61.8%
	FSG	66.2%	65.3%	62.3%	62.8%	64.5%
	SS-1	67.9%	68.4%	68.3%	68.0%	69.7%
	SS-2	69.1%	68.2%	68.8%	68.8%	69.8%
	DFDM	70.1%	71.4%	71.4%	73.7%	75.5%
	PMC (%)	9.3%	9.1%	8.8%	8.7%	8.6%
20-Class Experiments	**Adaboost**	23.3%	27.0%	27.0%	27.0%	27.0%
	Rotation Forest	47.7%	49.5%	49.5%	49.6%	50.4%
	Random Subspace	48.3%	48.1%	48.1%	48.6%	48.7%
	FSG	52.4%	50.7%	49.9%	50.9%	52.9%
	SS-1	54.0%	53.3%	53.8%	53.8%	56.2%
	SS-2	54.8%	53.2%	53.1%	53.2%	56.3%
	DFDM	54.9%	55.1%	55.1%	56.2%	57.3%
	PMC (%)	13.8%	13.1%	12.7%	12.7%	12.5%

In the second set of the experiments, the samples belonging to randomly selected classes are classified. Average (Ave.) and variance (Var.) of the classification performances of the benchmark and the proposed algorithms are given in Table 3.7 and Table 3.8, respectively. The classification results in the tables are depicted in Figure 3.5.

In the experiments, the difference between the classification performances of the benchmark and the proposed algorithms increases as the number of classes (C) increases. The performance of the Adaboost algorithm decreases faster than the other algorithms as C increases (see Figure 3.5). Moreover, the Adaboost algorithm performs better than the other benchmark algorithms for classifying the samples belonging to $C \leq 5$. However, the difference between the performance of the Adaboost and the other benchmark algorithms decreases for the classification of $C \geq 5$. Moreover, the performance of the Adaboost and the FSG is approximately same for $C = 2$ class classification. Note also that the difference between their performances increases as C increases. In addition, 1-NN classifier outperforms the Adaboost and is competitive to the other benchmark classifiers for $C \geq 7$.

Table 3.7: Classification results of the benchmark algorithms on the Corel Dataset.

	Benchmark Algorithms							
C	**Adaboost**		**RF**		**RS**		**1 NN**	
	Ave.	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**
2	90.56%	9.30%	86.00%	0.97%	88.11%	0.75%	82.44%	2.78%
3	81.33%	0.97%	76.27%	0.57%	75.87%	0.62%	75.27%	0.55%
4	73.45%	0.54%	69.75%	0.81%	70.45%	1.27%	69.60%	1.10%
5	64.32%	0.32%	62.72%	0.78%	65.32%	0.92%	61.08%	0.65%
6	61.17%	0.86%	61.67%	0.83%	64.20%	1.24%	60.50%	0.84%
7	54.12%	0.67%	58.00%	0.51%	62.98%	0.45%	56.98%	0.55%
8	53.17%	0.12%	60.03%	0.30%	54.92%	2.36%	58.22%	0.35%
9	49.02%	1.35%	56.98%	1.81%	55.89%	3.37%	54.98%	1.87%
10	39.65%	0.65%	48.35%	0.27%	47.00%	0.35%	47.60%	0.58%
12	38.64%	0.65%	45.57%	0.87%	43.22%	1.13%	45.02%	0.86%
14	33.16%	0.66%	47.16%	0.63%	46.81%	0.71%	45.76%	0.85%
16	29.54%	0.17%	40.42%	0.24%	41.53%	0.29%	39.86%	0.31%
18	25.30%	0.59%	41.56%	0.42%	40.91%	0.47%	39.97%	0.44%
20	19.46%	0.14%	38.27%	0.16%	39.98%	0.21%	36.25%	0.24%
25	16.15%	0.23%	35.92%	0.42%	35.57%	0.63%	33.94%	0.37%
30	14.37%	0.55%	33.53%	0.22%	36.28%	0.58%	32.43%	0.26%

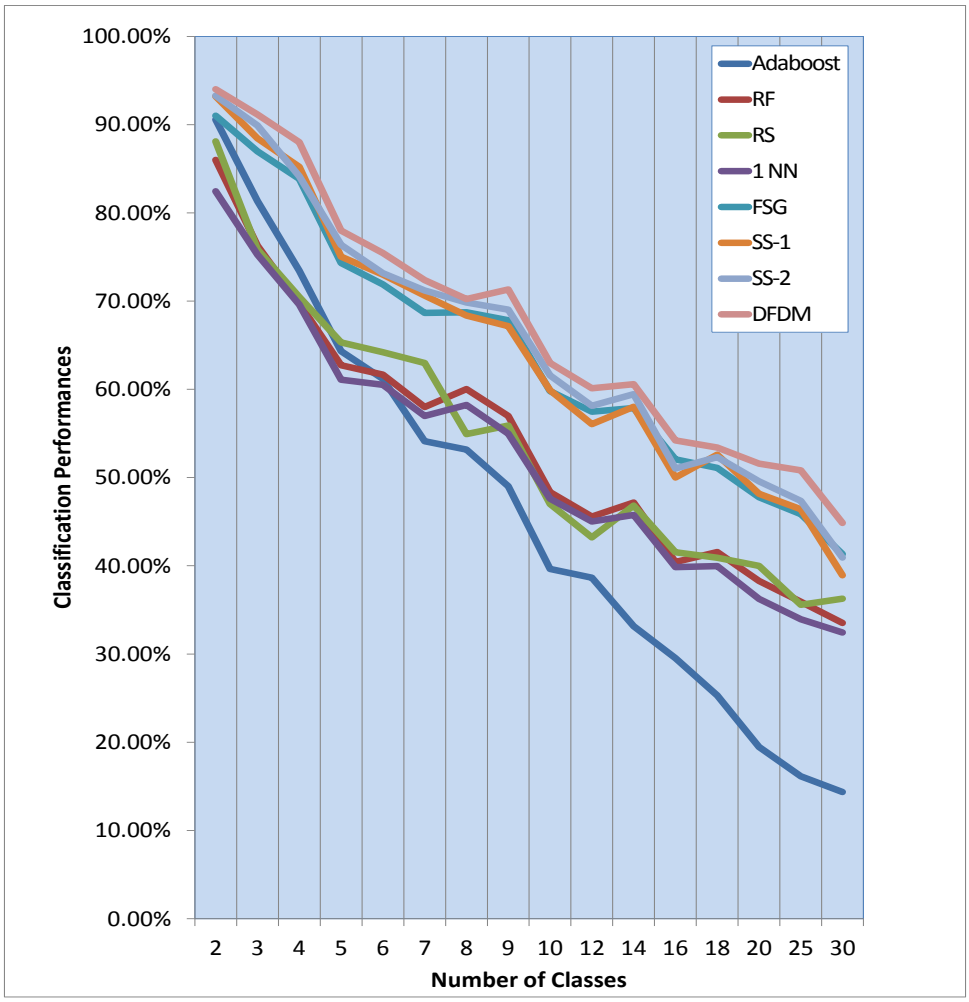


Figure 3.5: Classification performances of the algorithms on the Corel Dataset. Note that the best performance is achieved by the suggested Decision Fusion with Decision Margin (DFDM) algorithm.

Table 3.8: Classification results of the proposed algorithms on the Corel Dataset.

	Decision Fusion Algorithms							
C	**FSG**		**SS-1**		**SS-2**		**DFDM**	
	Ave.	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**
2	91.00%	0.43%	93.20%	0.41%	93.26%	0.41%	94.00%	0.34%
3	86.97%	0.53%	88.45%	0.53%	89.91%	0.53%	91.15%	0.55%
4	83.85%	0.59%	85.22%	0.63%	84.15%	0.60%	88.01%	0.56%
5	74.32%	0.42%	75.03%	0.48%	76.37%	0.45%	77.98%	0.45%
6	71.90%	0.67%	73.00%	0.77%	73.16%	0.84%	75.45%	0.71%
7	68.65%	0.44%	70.62%	0.52%	71.20%	0.56%	72.37%	0.43%
8	68.72%	0.28%	68.35%	0.33%	69.84%	0.35%	70.23%	0.19%
9	67.82%	1.16%	67.15%	1.57%	69.02%	1.66%	71.32%	1.22%
10	59.80%	0.37%	59.88%	0.75%	61.58%	0.58%	62.98%	0.61%
12	57.46%	0.48%	56.06%	0.64%	58.13%	0.69%	60.13%	0.50%
14	57.87%	0.75%	58.01%	1.12%	59.44%	0.98%	60.59%	0.72%
16	52.07%	0.44%	50.00%	0.45%	51.03%	0.67%	54.21%	0.37%
18	51.09%	0.47%	52.60%	0.45%	52.31%	0.42%	53.41%	0.40%
20	47.77%	0.20%	48.17%	0.31%	49.56%	0.25%	51.58%	0.18%
25	45.84%	0.42%	46.41%	0.72%	47.35%	0.66%	50.81%	0.35%
30	41.33%	0.52%	38.93%	0.92%	40.93%	0.89%	44.85%	0.69%

3.7.2.3.2 Experiments on Caltech Dataset

In Caltech Dataset experiments, the samples belonging to 2 to 10 different classes are randomly selected from the dataset. The experiments are repeated 10 times for each class.

In the experiments, the features provided by Gehler and Nowozin [123] are used for the construction of the feature spaces. Four feature spaces are constructed using three visual descriptors. Two features spaces consist of SIFT features extracted on a gray scale and an $\mathcal{HSI}$ image. The third and the fourth feature spaces contain the features extracted using Region Covariance and Local Binary Patterns descriptors. Implementation details of the features are given below [123].

- **SIFT** (D_j = 300 dimensional) descriptors model scale invariant appearance information [130]. In the experiments, the descriptors are computed in a regular grid on the image with a spacing of 10 pixels and four different radii $4, 8, 12, 16$. The descriptors are subsequently quantized into a vocabulary of visual words that is generated by k-means clustering. Two variants of the descriptor are used by implementing SIFT on a grey scale image (D_j = 128) and in $\mathcal{HSI}$ space (D_j = 384) of the image with a codebook of size 300.

- **Region Covariance** (D_j = 112) captures the statistical information on the features inside an image region by a covariance matrix of the features in that region [131]. Since covariance matrices do not lie on Euclidean space, they are projected to the tangent-space of a manifold Sym_7^+ which is a set of 7×7 dimensional symmetric positive definite matrices, i.e. a connected Riemannian manifold. The features are extracted at different scales of an image. The base scale is the image itself and the image is split into four

non-overlapping windows in each subsequent level. The features computed at the second level are used as the final feature.

- **Local Binary Patterns** (D_j = 148) extracts the local relationship between the pixels in a circular region by comparing the signal intensity value of a center pixel with that of its neighbor pixels in the region [132]. In the experiments, histograms of uniform rotation invariant $LBP_{8,1}$ features are computed using the gray values of 8 equally spaced pixels on a circle of radius 1 at the second level of the image scale representation described above.

Table 3.9: Classification results of the benchmark algorithms on the Caltech Dataset.

Benchmark Algorithms								
C	**Adaboost**		**RF**		**RS**		**1 NN**	
	Ave.	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**
2	96.47%	0.13%	87.72%	2.86%	87.70%	1.31%	87.78%	2.00%
3	89.68%	0.11%	80.90%	0.46%	81.20%	0.33%	80.90%	0.46%
4	81.21%	1.55%	74.17%	1.82%	76.10%	1.73%	72.20%	2.62%
5	83.27%	0.95%	77.66%	0.92%	76.91%	1.07%	77.55%	1.24%
6	85.14%	0.69%	82.73%	0.47%	83.42%	0.51%	80.97%	0.97%
7	77.00%	0.55%	76.86%	0.32%	76.79%	0.49%	76.71%	0.25%
8	68.49%	1.14%	71.46%	0.97%	70.13%	1.07%	66.77%	2.83%
9	75.48%	0.88%	75.90%	0.71%	75.93%	0.83%	75.69%	0.76%
10	64.30%	0.34%	65.66%	0.20%	65.47%	0.18%	62.30%	0.30%

Table 3.10: Classification results of the proposed algorithms on the Caltech Dataset.

Decision Fusion Algorithms								
C	**FSG**		**SS-1**		**SS-2**		**DFDM**	
	Ave.	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**	**Ave.**	**Var.**
2	95.64%	0.28%	96.15%	0.29%	97.42%	0.19%	99.05%	0.15%
3	90.46%	0.12%	91.07%	0.12%	92.33%	0.18%	94.01%	0.22%
4	85.32%	0.70%	85.75%	0.81%	86.40%	0.74%	87.93%	1.19%
5	88.57%	0.41%	89.47%	0.69%	90.09%	0.70%	91.26%	0.61%
6	92.15%	0.25%	93.89%	0.33%	94.88%	0.21%	95.49%	0.85%
7	88.54%	0.23%	87.93%	0.99%	88.90%	0.50%	90.79%	0.25%
8	85.89%	0.35%	86.17%	0.14%	86.31%	0.95%	87.09%	0.99%
9	86.28%	0.24%	87.20%	0.41%	88.27%	0.44%	89.05%	0.39%
10	81.06%	0.23%	82.20%	0.33%	83.01%	0.29%	84.67%	0.85%

In the experiments with Caltech dataset, classification performances of the algorithms do not decrease linearly by increasing number of classes as observed in the experiment with Corel dataset. Note that this non-linear performance variation is observed for all of the aforementioned algorithms. This may be occurred because of the uncertainty in the feature vectors of the samples which is caused by the descriptors employed on the dataset instead of the instability of the classification algorithms. For instance, Adaboost performs better than the other algorithms for C = 2. In addition, the difference between the classification performances of Adaboost and FSG is 0.78% for C = 3. However, the difference increases to

17.40% for C = 8 where Adaboost performs worse than the other algorithms. In all of the experiments, DFDM outperformed the other algorithms.

3.8 Chapter Summary

In this chapter, the classification error minimization problem have been addressed for Decision Fusion. The error minimization problem has been studied using k-Nearest Neighbor algorithm in two parts; i) the minimization of the difference between N-sample and large-sample classification errors, and ii) the minimization of the large-sample classification error.

k-NN algorithm has been employed for the analysis of the error difference minimization problem because of three reasons. First, the error of k-NN is upper and lower bounded by the Bayes Error which is the minimum achievable classification error by any classification algorithm. Therefore, the error bounds are tractable. Second, k-NN can be considered as a decision fusion algorithm which combines the decisions of the neighbor samples of a given test sample to estimate its label or class membership value. In addition, k-NN is a *powerful* nonparametric density estimation algorithm used for the estimation of posterior probabilities to design distance functions.

Distance learning problem for classification error minimization is analyzed as a feature vector mapping, decision and feature space design, i.e. classifier design and decision fusion, problem. In order to solve these problems, a hierarchical decision fusion algorithm called Fuzzy Stacked Generalization (FSG) is employed.

Base-layer classifiers of the FSG are used for two purposes; i) mapping feature vectors to decision vectors and ii) estimating posterior probabilities, which are the variables of the distance function, using the datasets. Decision vectors in the decision spaces are concatenated to construct the feature vectors in the fusion space for a meta-layer classifier.

One of the major contributions of the suggested decision fusion methods is to minimize the difference between N-sample and large-sample classification error of k-NN. This property is shown by using the distance learning approach of Short and Fukunaga [14]. In addition, samples should be classified with complete certainty as shown by Cover and Hart [13] in order to converge the large-sample classification error to Bayes Error. In FSG, this condition should be satisfied by the meta-layer classifier employed on the fusion space in almost everywhere, i.e. by each feature vector of each sample.

In order to assure this condition, first a distance function called Decision Margin (DM), which measures the distance between the feature of a sample and its target location defined by its target label in the fusion space, is introduced. Moreover, DM gives information about the decisions of base-layer classifiers on the samples. Therefore, the relationship between classification performances of base-layer classifiers and meta-layer classifier can be analyzed using DM.

Minimization of DM values of the samples in decision spaces can be achieved by minimizing the N-sample classification errors of the base-layer classifiers. Moreover, large-sample error of the meta-layer classifier can be minimized by minimizing DM values in the fusion space. For instance, if a sample is correctly classified by base-layer classifiers with maximum certainty, then DM of the sample is minimum. If DM of the sample is minimum, then the feature

vector of the sample is closer to the vertex which represents its target class label than the other vertices of a polytope of feature vectors in the fusion space. If DM values of all of the samples are minimum, then the samples are correctly classified with maximum certainty by the meta-layer classifier.

In order to compute the weights which minimize DM values, weighted decision fusion problem is defined as a convex optimization problem. In order to employ a decision selection strategy, which assigns zero weights to decisions of base-layer classifiers that provide decisions with *large* DM values, sparsity constraints are used in the optimization problem. Then, the optimization problem is solved using an optimization algorithm called Alternating Direction Method of Multipliers.

Decision Fusion with Decision Margin (DFDM) algorithm minimizes DM values of samples averaged over the training dataset. However, we may also need to assure that each sample is correctly classified with complete certainty in the fusion space. This requirement can be partially satisfied by forcing the samples that belong to the same class in the same Voronoi region in the fusion space. Noticing that the decision vectors of the samples provided by the base-layer classifiers define the belongingness of the samples to Voronoi regions, the features of the samples that could not be correctly labeled by at least one base-layer classifier are eliminated from the base-layer and meta-layer feature spaces using two sample selection algorithms, SS-1 and SS-2, respectively.

The proposed algorithms are tested on artificial and benchmark datasets by the comparisons with the state of the art algorithms such as Adaboost, Rotation Forest and Random Subspace.

In the first group of the experiments, the samples belonging to different classes are gradually overlapped in the synthetic datasets. The experiments are designed in such a way that the requirements of N-sample and large-sample error minimization conditions are controlled.

It is observed that if one can design the feature spaces such that $\hat{Ave}_{corr} \approx 1$, the classification performance of FSG becomes significantly higher than that of the individual classifier performances. This experiment also shows that the performance of FSG depends on sharing and collaborating the features of the samples rather than the performance of individual classifiers.

In the experiments, sample selection methods boost the performance of FSG if PMC is not too *large* such that the removal of the samples from the training datasets does not cause singularities in the datasets. In addition, eliminating the misclassified samples from the base-layer output space may bring many problems, such as loss of information or curse of dimensionality. In all of the experiments, DFDM has provided better classification performances than the other algorithms.

In the second group of experiments, the proposed algorithms are compared with state of the art algorithms using benchmark datasets. In two class multi-attribute classification experiments, FSG and Adaboost provide similar performances. Meanwhile, Adaboost outperforms FSG for the classification of samples with high dimensional features such as $D = 20$. This is observed due to the curse-of-dimensionality observed in the fusion space which is $20 \times 2 = 40$ dimensional. However, DFDM eliminates decisions of some of the base-layer classifiers in the construction of the fusion space, therefore, provides better classification performance.

FSG with sample selection and DFDM algorithms outperform state of the art algorithms in the experiments employed on multi-feature datasets, especially for the classification of the samples

belonging to $C > 2$ number of classes. The difference between the classification performances of the proposed algorithms and the state of the art algorithms employed on multi-feature datasets is greater than that of the difference observed in multi-attributed datasets because of two reasons. First, the proposed algorithms fix the dimensions of the feature vectors in fusion space to CJ (number of classes × number of feature extractors) no matter how high is the dimension of the individual feature vectors at the base-layer. Second, employing distinct feature extractors for each base-layer classifier enables us to split various attributes of the feature spaces, coherently. Therefore, each base-layer classifier gains an expertise to learn a specific property of a sample, and correctly classifies a group of samples belonging to a certain class in the training data. This approach assures the diversity of the classifiers as suggested by Kuncheva [133] and enables the classifiers to collaborate for learning the classes or groups of samples. It also allows us to optimize the parameters of each individual base-layer classifier independent of the other.

CHAPTER 4

APPLICATIONS OF DECISION FUSION ALGORITHMS

4.1 Building Detection with Decision Fusion

Building detection is one of the challenging problems of target detection in remote sensing applications. Although various algorithms have been developed for a better automatic building extraction, they do not provide an exact solution for an automated building detection system. The state of the art methods aim to solve this problem with a great variety of approaches, which can be grouped under unsupervised and supervised detection methods. The unsupervised algorithms basically, detect the buildings using predefined rule-based models and unsupervised classifiers. A popular approach in this group is to employ the shape based features which represent the rectangular structure of the roofs [134].

Some of the unsupervised learning algorithms employ techniques to discriminate and remove the irrelevant regions from the image. Then, they focus on the regions which include buildings. Sirmacek and Unsalan [135] formulate the urban-region and building detection problems using graphical models. Ok *et al.* [136] use the shadow evidence to focus on building regions. Their proposed approach models the directional spatial relationship between buildings and their shadows.

A group of unsupervised methods is supported by a set of knowledge based rules to compute decisions [134, 137]. The main drawback of this model is the construction of knowledge based rules which are defined using object specific assumptions such as the size or shape of the buildings. Therefore, the performance of the algorithm depends on the validity of the assumptions; if the assumptions are not valid for the test images, then the algorithm may fail. For instance, Saeedi and Zwick [134] assume that the buildings are separated by a distance value which is smaller than a user defined threshold value. However, the threshold value depends on the type and the location of the buildings, such that the hotels in urban areas and the cottages in rural areas are separated by different distances.

In supervised learning approaches, a set of training samples (i.e., pixels, segments or features with ground truth data) is used to train the classifiers which are employed to classify a given test sample. The crucial step of the supervised building detection methods is the design of the feature spaces. A popular approach is to use the mean color of the RGB bands and simple texture features. A good example of this approach can be found in Turker and Koc San [138], where they use color intensity, normalized digital surface model and Normalized Difference Vegetation Index (NDVI) for feature extraction. Also, Inglada [139] extracts low and high

level geometric features which are aggregated under the same vector space. Recent studies on building detection methods are reviewed in [140, 141]. Mountrakis *et al.* suggest to use another state of the art supervised learning algorithm Support Vector Machines (SVMs) [142] in remote sensing applications [143].

In this study, building detection problem is solved by the decision fusion algorithm Fuzzy Stacked Generalization (FSG) introduced in Chapter 3. Figure 4.1 shows the block diagram representation of the suggested building detection method. In the first step of the method, a remotely sensed image is segmented using Mean Shift segmentation method [144]. In order to optimize the parameters of the Mean Shift algorithm, a new metric called Overall Segmentation Quality is defined, and an optimal parameter set which maximizes this metric is selected for each experiment. Then, the segments which belong to the vegetation and shadow regions are identified and discarded from the image. Next, various multi-modal color, shape and texture features are extracted from each segment. The features are fed to separate classifiers. Finally the output decision spaces of each classifier are fused in FSG to detect buildings.

The performance of the proposed algorithm is examined using a multi-spectral image dataset which consists of QuickBird satellite image with a 60cm spatial resolution. The performance of FSG is compared with the other machine learning algorithms used for building detection, such as SVMs and with one of the recent approaches proposed by Turker and Koc San mentioned above [138].

The next subsection describes the proposed building detection algorithm which employs the FGS.

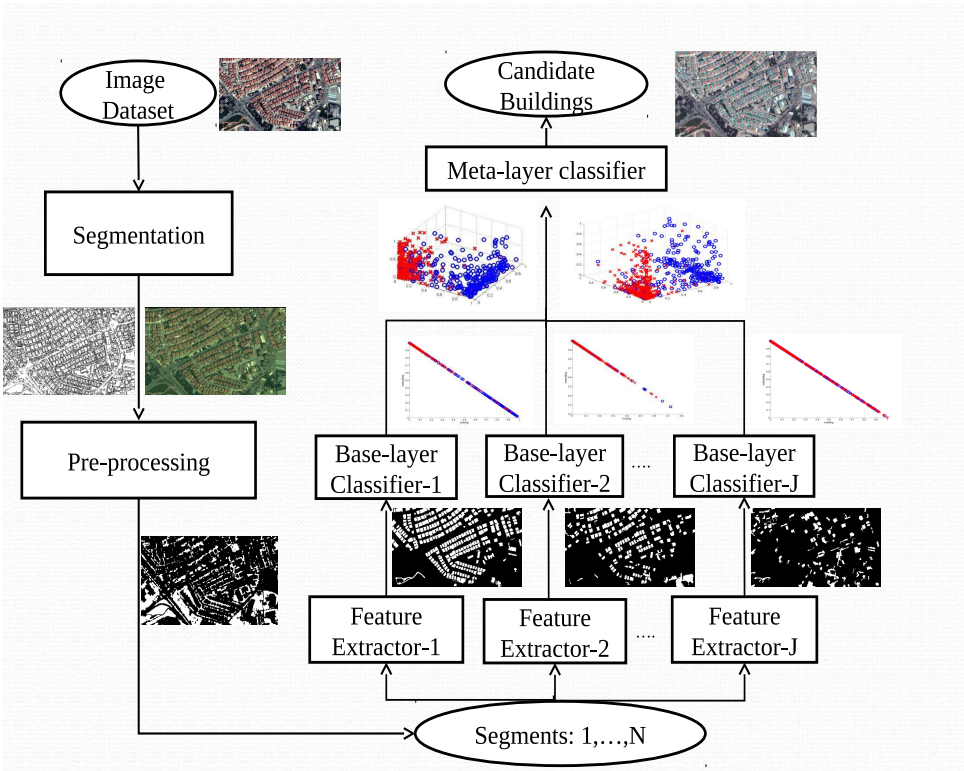


Figure 4.1: The building detection method with decision fusion.

4.1.1 An Overview of the Proposed Building Detection Method

This section introduces the major steps of the suggested building detection algorithm, which includes, segmentation, pre-processing of the segments using vegetation and shadow elimination, feature extraction and hierarchical classification with FSG.

4.1.1.1 Mean Shift Segmentation and Optimization of its Parameters

In this chapter, Mean Shift segmentation algorithm [144] is used for segmenting the remotely sensed images. The Mean Shift algorithm is a density estimation based mode detection and data clustering algorithm, which was proposed by Fukunaga and Hostetler [145]. The algorithm has been used successfully for image segmentation and clustering [144].

Given a set of vectors $\bar{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, N$, with bandwidth values $h_i > 0$, the estimator is

$$\hat{f}_K(\bar{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h_i^D} k\Big(\left\| \bar{x} - \bar{x}_i \right\|^2 \Big),$$

where $k(\cdot)$ is called the profile of the spherically symmetric kernel K defined as

$$K(\bar{x}) = c_{k,D} k\Big(\|\bar{x}\|^2 \Big) ,$$

where $K(\hat{x}) > 0$, $\|\hat{x}\| \leq 1$ and $c_{k,D}$ is a normalization constant which assures the integration of $K(\hat{x})$ to 1.

In Mean Shift, since we are interested in the differentials of the kernels, we seek the modes and the means of the distributions in the algorithm. Therefore, the differential of the kernel profile is defined by the function

$$g(\bar{x}) = -k^{'}(\bar{x}),$$

when the differential of $k(\bar{x})$ exists.

When the gradient of the estimator is computed, we get the following mean shift vector:

$$\hat{\bar{x}} = \frac{\sum_{i=1}^{N} \bar{x}_i \frac{1}{h_i^{(D+2)}} g\Big(\| \frac{\bar{x} - \bar{x}_i}{h_i} \|^2 \Big)}{\sum_{i=1}^{N} \frac{1}{h_i^{(D+2)}} g\Big(\| \frac{\bar{x} - \bar{x}_i}{h_i} \|^2 \Big)} - \bar{x} ,$$

which points towards the direction of maximum increase in the density [144].

Then, we achieve the nearest stationary point of the density iteratively via

$$\hat{\bar{x}}_{j+1} = \frac{\sum_{i=1}^{N} \bar{x}_i \frac{1}{h_i^{(D+2)}} g\Big(\| \frac{\hat{\bar{x}}_j - \bar{x}_i}{h_i} \|^2 \Big)}{\sum_{i=1}^{N} \frac{1}{h_i^{(D+2)}} g\Big(\| \frac{\hat{\bar{x}}_j - \bar{x}_i}{h_i} \|^2 \Big)} , \; j = 1, 2, \ldots .$$

In the employment of the Mean Shift algorithm to image segmentation, the spatial coordinates are evaluated with the feature vectors. Therefore, the density is estimated in a joint domain and the density estimation kernel is defined as the product of two radially symmetric kernels with a single bandwidth parameter for each domain as follows,

$$K(\bar{x}) = \frac{C}{h_s^2 h_r^{D-2}} k\Big(\Big\|\frac{\bar{x}^s}{h_s}\Big\|^2\Big) k\Big(\Big\|\frac{\bar{x}^r}{h_r}\Big\|^2\Big)\,, \tag{4.1}$$

where h_s is the spatial resolution parameter, which affects the smoothing and the connectivity of the segments, and h_r is the range resolution parameter which affects the number of the segments [144]. $\bar{x}^s$ represents the spatial coordinate of the pixels and $\bar{x}^r$ represent the range of the pixels, such that $\bar{x}^r \in \mathbb{R}^3$ for the segmentation of an RGB image. Some of the *smallest* regions are eliminated or merged with the nearest regions in order to deal with the noisy patches in the implementations. The size of the smallest region is defined by the user with the parameter $minArea$.

In the employment of the Mean Shift algorithm to image segmentation in this chapter, R, G and B pixel values are transformed into LUV color space. A fixed kernel, with bandwidth parameters h_s and h_r is constructed. Then, the kernel is shifted to the point corresponding to the mean value of the sample points (feature values of the pixels in LUV space) which are within the corresponding bandwidth of each sample point. This process is iteratively applied until the Mean Shift vector converges to a point at which the density estimations are stationary, i.e. do not change. After the convergence is achieved, the pre-segments which are within the pre-defined spatial and spectral proximity, are merged. Finally, the *smallest* regions are eliminated or merged with the nearest regions to compensate for the noisy patches. The size of the smallest region is defined by the user with the parameter $minArea$, as will be explained in Section 4.1.2.

The detection performance of the buildings highly depends on the segmentation output. In other words, the features extracted from the segments, which represent the buildings in a single region are more discriminative than the features extracted from the output of under-segmented or over-segmented images. Although there are several studies focused on the selection of *optimal* Mean Shift segmentation parameters [146],[147], in this study, a computationally efficient approach which is designed for building detection problem, is suggested to estimate the *optimal* parameters.

Let $S_{\psi_j} = \{s_i\}_{i=1}^{N}$ be the segmentation result which is obtained from the Mean Shift algorithm implemented with a parameter tuple $\psi_j = (hs_j, hr_j, minArea_j)$ selected from a set of parameters $\psi = \{\psi_j\}_{j=1}^{\Lambda}$ for a training image, $\mathbb{I}$. The pixels of $\mathbb{I}$ are marked as building or background using a ground truth image, $\mathbb{I}^G$. The number of pixels in a segment s_i is defined as M_i, and the number of pixels which represent buildings (i.e., the pixels which are marked as buildings in $\mathbb{I}^G$) in s_i is defined as M_i^b. Then, a set of segments is defined as $D_{\psi_j}^b = \{s_i : M_i^b > \mathfrak{s} M_i\}$, where $\mathfrak{s} \in \mathbb{R}$ is a given constant. In order to select the optimal ψ_j parameters, two metrics are defined as

$$SF_{\psi_j} = \frac{\left|\mathcal{M}(D_{\psi_j}^b) \bigcap \vartheta^G\right|}{\left|\mathcal{M}(D_{\psi_j}^b)\right|}, \tag{4.2}$$

$$GTC_{\psi_j} = \frac{\left|\mathcal{M}(D_{\psi_j}^b) \bigcap \vartheta^G\right|}{|\vartheta^G|}, \tag{4.3}$$

where $\mathcal{M}(\cdot)$ defines a set of pixels in the boundaries of segments in $D_{\psi_j}^b$. ϑ^G is the set of the boundary pixels in the ground truth image, and $|\cdot|$ is the set cardinality. SF_{ψ_j} (Segmentation Fitting) measures the accuracy of individual segments to represent a building, and GTC_{ψ_j}

(Ground Truth Coverage) measures the quality of representing a building's boundary without considering the number of oversegments. Finally, the Overall Segmentation Quality (OSQ_{ψ_j}) is computed by combining these metrics under one equation as follows;

$$OSQ_{\psi_j} = 2\frac{\left(SF_{\psi_j}\right)\left(GTC_{\psi_j}\right)}{SF_{\psi_j} + GTC_{\psi_j}}. \tag{4.4}$$

Then, an optimal parameter tuple $\hat{\psi}$ is selected using

$$\hat{\psi} = \underset{\psi_j \in \psi}{\operatorname{argmax}}\, OSQ_{\psi_j}. \tag{4.5}$$

In the experiments, (4.5) is solved by exhaustive search on a given set of parameters ψ. The implementation details for finding an optimal parameter tuple $\hat{\psi}$ are given in the experiments.

4.1.1.2 Pre-processing: Elimination of Vegetation and Shadow Regions

The segments which belong to vegetation and shadow regions are identified and discarded to avoid the false alarms in building detection. In order to detect vegetation regions, Normalized Difference Vegetation Index (NDVI) is utilized [148] as

$$NDVI = \frac{NIR - R}{NIR + R}, \tag{4.6}$$

where R and NIR represent red and near-infrared bands, respectively. For each image, a threshold value τ_{NDVI} is computed using the Otsu method [149]. Then, the pixels whose $NDVI$ values are greater than τ_{NDVI} are marked as vegetation pixels, and a vegetation mask is generated. The number of vegetation pixels in each segment s_i is defined as M_i^v. Then, a segment s_i is labeled as vegetation, if $M_i^v > \mathfrak{s} M_i$.

Moreover, a multi-spectral false color shadow detection algorithm proposed by Teke *et al.* [150] is used due to two important reasons:(i) their proposed approach utilizes the advantage of the NIR band, and (ii) the shadow detection algorithm does not require user defined thresholds. In this approach, NIR, R and G are used to generate a false color image. Once the false color image (NIR-R-G) is obtained, it is converted into $\mathcal{HSI}$ color space. The $RatioMap$ is defined as

$$RatioMap = \frac{\mathcal{S} - \mathcal{I}}{\mathcal{S} + \mathcal{I}}, \tag{4.7}$$

where $\mathcal{S}$ is the normalized saturation, and $\mathcal{I}$ is the normalized intensity. The extracted $RatioMap$ is binarized using Otsu method to obtain a set of pixels which represent both shadows and vegetations, called shadow and vegetation pixels, respectively. After the vegetation pixels are eliminated using $NDVI$, a segment s_i is labelled as a shadow segment, if $M_i^s > \mathfrak{s} M_i$, where M_i^s is the number of shadow pixels in s_i.

4.1.1.3 Features used in the Building Detection Algorithm

Given a segmented image represented by the dataset $S_{\hat{\psi}} = \{s_i\}_{i=1}^N$ consisting of N segments s_i, a set of features F_j is constructed using the following j^{th} feature extractor;

$$S_{\hat{\psi}} = \{s_i\}_{i=1}^N \longrightarrow F_j = \{\bar{x}_{i,j}, y_i\}_{i=1}^N, \forall j = 1, 2, \ldots, J, \tag{4.8}$$

where $\bar{x}_{i,j} \in \mathbb{R}^{D_j}$ is the feature vector extracted from the i^{th} segment in the dataset, and y_i is the label of the corresponding segment. The base-layer consists of J classifiers each of which is fed by a set of distinct features extracted from the same segment. A feature set F_j, which is fed to an individual base-layer classifier is selected to represent different physical properties of the segments. Therefore, the feature extractors are considered as low-level information extractors.

The features employed in this section have been studied in various papers on object detection [151] [81]. Since the goal of this section is to show the performance improving effect of the suggested Fuzzy Stacked Generalization architecture, feature or sample selection algorithms have not been implemented for this problem.

A diverse set of features, which are used to extract information about color, texture and shape characteristics of segments, is given below. Mathematical definitions of the features which are used to construct F_j and the dimensions of the feature vectors are provided in Table 4.1. The dimensions of the feature vectors described in this section, are given in Table 4.1.

- **Color Features** For each segment s_i, the standard deviation of the intensity values of the pixels in s_i, $stdc_i \in \mathbb{R}^{D_{im}}$, is computed, where $D_{im} = 4$ is the number of color bands. Moreover, each color band is divided into 8 histogram bins for each segment $s_i, \forall i = 1, 2, \ldots, N$. Then, a probability density vector ($hist\_pdv_i$), which describes the ratio of the number of pixels belonging to each bin to the total number of pixels, is computed for each segment s_i. In addition, for each s_i, the first ($hist\_mean_i$), the second ($hist\_variance_i$), the third ($hist\_skewness_i$) and the fourth ($hist\_kurtosis_i$) moments, $hist\_energy_i$ and $hist\_entropy_i$ measures are computed using the probability density values estimated from the histogram, as described in [152], and given in Table 4.1.

- **Shape Features** Three shape features are extracted for each segment. The first feature is the $area_i$, which is the number of pixels belonging to s_i. In order to utilize the rectangular shape properties of the buildings, $rectangularity_i$ is computed. Moreover, the major axis length and minor axis length are concatenated to generate a two-dimensional shape vector, $axis\_lengths_i$.

- **Texture Features** Texture provides important information on the structure of the objects in remote sensing. In this study, the Gray-Level Co-occurrence Matrix (GLCM) [153] is used to extract textural information. In order to construct a GLCM, each band is employed as a grayscale image, and discretized into 8 levels. After discretization, 4 different GLCMs of each segment s_i are computed in four directions $\{0, 45, 90, 135\}$, and then these matrices are used to compute $glcm\_contrast$, $glcm\_correlation$, $glcm\_energy$, $glcm\_entropy$ features as suggested in [153].

4.1.1.4 Decision Fusion with the Fuzzy Stacked Generalization

Building detection problem is formulated as a two-class classification problem in which the segments with $y_i = 0$ belong to background class, and the segments with $y_i = 1$ belong to building class. The membership values, $\mu_c(\bar{x}_{i,j})$ of the features $\bar{x}_{i,j}$ for the c^{th} class, $\forall c = 1, 2$, are computed by each base-layer classifier using (3.9). Therefore, a two-dimensional

Table 4.1: Mathematical definitions of the feature extraction algorithms. Recall that, F_j represents the j^{th} feature space, D_j represents the dimension of F_j, where a different descriptor is used in each F_j.

F_j	D_j	**Description**
$stdc_i = \left[stdc_i^1, stdc_i^2, stdc_i^3, stdc_i^4\right]$	4	$stdc_i^{D_b} = \sqrt{\frac{1}{M_i}\sum_{j=1}^{M_i}(x_{ij}^{D_b} - mc_i^{D_b})^2}$, where $mc_i^{D_b} = \frac{1}{M_i}\sum_{j=1}^{M_i} x_{ij}^{D_b}$, D_b is the selected band and M_i is the number of pixel in s_i and x_{ij} is the intensity value of j^{th} pixel in s_i
$hist\_pdv_i = \left[pdv_{i1}^1, pdv_{i2}^1, \ldots, pdv_{i8}^4\right]$	32	$pdv_{im}^{D_b} = \frac{h_{im}^{D_b}}{\sum_{n=1}^{8} h_{in}^{D_b}}$, where h_{im}^d is the number of elements in m^{th} bin which is generated from ${D_b}^{th}$ band of s_i.
$hist\_m_i = [hm_i^1, hm_i^2, hm_i^3, hm_i^4]$	4	$hm_i^{D_b} = \sum_{n=1}^{8} pdv_{in}^d n$
$hist\_v_i = \left[hv_i^1, hv_i^2, hv_i^3, hv_i^4\right]$	4	$hv_i^{D_b} = \frac{1}{N}\sum_{n=1}^{8}(pdv_{in}^{D_b}(n - hist\_m_i^{D_b})^2)$
$hist\_s_i = \left[hs_i^1, hs_i^2, hs_i^3, hs_i^4\right]$	4	$hs_i^{D_b} = \frac{1}{\sqrt{hist\_var_i}^3}\sum_{n=1}^{8}(pdv_{in}^{D_b}(n - hist\_m_i^{D_b})^3)$
$hist\_k_i = \left[hk_i^1, hk_i^2, hk_i^3, hk_i^4\right]$	4	$hk_i^{D_b} = \frac{1}{\sqrt{hist\_var_i}^4}\sum_{n=1}^{8}(pdv_{in}^{D_b}(n - hist\_m_i^{D_b})^4)$
$hist\_energy_i = \left[hen_i^1, hen_i^2, hen_i^3, hen_i^4\right]$	4	$hen_i^{D_b} = \sum_{n=1}^{8}(pdv_{in}^{D_b})^2$
$hist\_entropy_i = \left[het_i^1, het_i^2, het_i^3, het_i^4\right]$	4	$het_i^{D_b} = \sum_{n=1}^{8} pdv_{in}^{D_b}\log(pdv_{in}^{D_b})$
$rectangularity_i = \frac{M_i}{major_i minor_i}$	1	$minor_i$ and $major_i$ are the length of major and minor axes lengths
$axis\_lengths_i = [major_i, minor_i]$	2	$minor_i$ and $major_i$ are the length of major and minor axes lengths
$area_i = M_i$	1	M_i is the number of pixel in s_i
$glcm\_contrast_i = \left[g\_ct_{0i}^1, \ldots, g\_ct_{135i}^4\right]$	16	$g\_ct_{\psi i}^d = \sum_{n=0}^{G-1} n^2\{\sum_{k=1}^{G}\sum_{\lvert j-k\rvert=n}^{G} p_{\psi i}^{D_b}(k,j)\}$, where $p_{\psi i}^{D_b}$ is the GLCM, generated from band the D_b, in ψ direction and G is the number of distinct gray levels in the quantized image.
$glcm\_correlation_i = \left[g\_cr_{0i}^1, \ldots, g\_cr_{135i}^4\right]$	16	$g\_cr_{\psi i}^{D_b} = \frac{\sum_k\sum_j (jk)p_{\psi i}^{D_b}(k,j) - \mathfrak{m}^2}{\sigma^2}$, where $\mathfrak{m}$ and σ are mean and variance
$glcm\_energy_i = \left[g\_ener_{0i}^1, \ldots, g\_ener_{135i}^4\right]$	16	$g\_ener_{\psi i}^d = \sum_k\sum_j p_{\psi i}^{D_b}(k,j)^2$
$glcm\_entropy_i = \left[g\_et_{0i}^1, \ldots, g\_et_{135i}^4\right]$	16	$g\_et_{\psi i}^{D_b} = -\sum_k\sum_j p_{\psi i}^{D_b}(k,j)\log(p_{\psi i}^{D_b}(k,j))$

membership vector is obtained for each segment s_i at the output of each base-layer classifier as

$$\bar{\mu}(\bar{x}_{i,j}) = \big[\mu_1(\bar{x}_{i,j}), \mu_2(\bar{x}_{i,j})\big]. \tag{4.9}$$

The above membership vector of each segment s_i carries information about the decisions and performances of the base-layer classifiers for identifying the class label of s_i with respect to its input feature vector $\bar{x}_{i,j}$. Decision vectors of the base-layer classifiers are aggregated to create the feature vectors of decision space. Then a meta-layer fuzzy k-NN classifier (3.9) is implemented using the feature vectors in the decision space for the detection of the buildings using FSG.

In the experiments, the performance of FSG is compared with the performance of the state of the art Support Vector Machines (SVM) classifier. Therefore, a brief description of SVM is given in the following subsection.

4.1.1.5 Support Vector Machines

The k-NN algorithm employs a non-linear classification rule in a feature space F_j. Alternatively, Support Vector Machines (SVMs) employ linear classification rules that separate the samples into two half spaces using hyperplanes in a high dimensional feature space, $\mathcal{F}_j$, which is constructed by a non-linear mapping $f : F_j \rightarrow \mathcal{F}_j$ [154]. The motivation to employ a mapping f is to construct linearly separable classes $\mathcal{F}_j$, where the classification rules exhibit higher performance than in F_j. Note that, a linear classifier employed in $\mathcal{F}_j$ is equivalent to a non-linear classifier in F_j.

In a D'_j dimensional feature space $\mathcal{F}_j$, a hyperplane is represented by a weight vector $\bar{w}_f \in \mathbb{R}^{D'_j}$ and a bias variable $b \in \mathbb{R}$, which results in

$$\bar{w}_f \cdot f(\mathbf{s}) + b = 0, \tag{4.10}$$

where $\bar{w}_f$ is normal to the hyperplane, and $f(\mathbf{s})$ is the feature vector of the sample that lies on the hyperplane [155]. That being the case, $\frac{|b|}{\|\bar{w}_f\|_2}$ is the perpendicular distance between the origin and the hyperplane. Note that $\mathcal{F}_j$ can be infinite dimensional, for instance, when mapping f is exponential, it results in Gaussian (RBF) kernels. In this case, the problem is formulated in dual-form. Therefore, the *kernel trick*, which enables the representation of the inner products between feature vectors in a kernel, can be used in order to compute the hyperplanes [156].

In our case, we seek a hyperplane which linearly separates the features of the samples that belong to the background and building classes in $\mathcal{F}_j$. Clearly, there are infinitely many hyperplanes that linearly separate these features. However, we also need to correctly classify a new test segment s'_i using the hyperplane. In order to satisfy this requirement, we choose the hyperplane that is at the largest distance from the closest feature vectors of the samples that belong to the background and building classes. For instance, if d_+ and d_- are the shortest distances from the hyperplane to the closest feature vectors of the samples from background and building classes, respectively, we look for the hyperplane with the maximum distance $d_+ + d_-$. This constraint can be formulated [155] as

$$y_i(\bar{w}_f \cdot f(\mathbf{s}) + b) - 1 \geq 0, \quad \forall i = 1, 2, \ldots, N. \tag{4.11}$$

Note that, $d_+ = d_- = \frac{1}{\|\bar{w}_f\|_2}$. Therefore, a maximum margin hyperplane can be computed by minimizing $\|\bar{w}_f\|_2$ and using quadratic optimization.

If the training examples in the transformed space are not linearly separable, the optimization problem can be modified by introducing slack variables $\xi_i \geq 0$, $\forall i = 1, 2, \ldots, N$, in (4.11) which yields

$$y_i(\bar{w}_f \cdot f(\mathbf{s}_i) + b) - 1 + \xi_i \geq 0 \, , \forall i = 1, 2, \ldots, N. \tag{4.12}$$

In order to compute the hyperplane given in (4.12), the optimization problem can be posed [32] as

$$\begin{aligned} \text{minimize} \quad & \|\bar{w}_f\|_2^2 + C \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(\bar{w}_f \cdot f(\mathbf{s}_i) + b) - 1 + \xi_i \geq 0 \\ & \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, N \end{aligned} \tag{4.13}$$

where C is a constant. This optimization problem is solved either in primal form [157] or in a dual form, which leads to a convex quadratic optimization problem [154, 156, 32], using various SVM algorithms. In the dual form, the problem is defined as

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} a_i - \tfrac{1}{2} \sum_{i,j} a_i a_j y_i y_j K(i,j) \\ \text{subject to} \quad & \sum_{i=1}^{N} a_i y_i = 0 \\ & C \geq a_i \geq 0, \quad \forall i = 1, 2, \ldots, N, \end{aligned} \tag{4.14}$$

where $K(i,j)$ is called the Kernel matrix. If $K(i,j) = f(\mathbf{s}_i) f(\mathbf{s}_j)$, i.e. the inner product of $f(\mathbf{s}_i)$ and $f(\mathbf{s}_j)$, an SVM which solves (4.14) is called SVM with Linear Kernel (*SVM-Linear*). If $K(i,j) = \exp(-\frac{1}{h_{svm}}^2 \parallel f(\mathbf{s}_i) - f(\mathbf{s}_j) \parallel_2^2)$, then an SVM which solves (4.14) is called SVM with Gaussian Kernel (*SVM-RBF*), where h_{svm} is the width of the kernel.

4.1.2 Experiments

In the experiments, a QuickBird satellite image is used. The image includes red, green, blue and near infra-red bands with a 60cm spatial resolution. 10 different patched images are selected from the whole image, and some of these images are illustrated in Figure 4.2. The sizes of four of the patched images are 1646×929, 1535×968, 1434×905, 1917×1004, and the sizes of the rest of the patched images are 1934×1013. Each image belongs to a different district of Ankara. During the performance evaluation, randomly selected 4 images are used for training, and the rest of 6 images are used for testing. Then, the average of the performance values of 10 experiments with different training subsets is computed. The reference buildings for each image are marked manually by a qualified human operator to generate the ground truth.

In this study, $\mathfrak{s}$ value is set to 0.5 which means, if at least a half of a segment overlaps with vegetation or shadow masks then this segment represents a vegetation or shadow region, respectively. Moreover, if at least a half of a segment overlaps with the reference buildings then this segment considered as a building during the performance calculations. Higher $\mathfrak{s}$

values result in higher precision, but lower recall values. In order to keep the balance between the precision and recall, $\mathfrak{s}$ value is set to 0.5.

The classifier decisions are grouped in four distinct categories as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) as given in Table 4.2. The segments which are detected as building, and also marked as building in the ground truth data are categorized as TP. On the other hand, if a building segment is not detected correctly, then the decision on the segment is categorized as FN. If the algorithm labels a non-building segment as a building, the decision on the segment is categorized as FP. Finally, the decisions on the segments, which are not detected as a building, and do not represent a building in the reference data, are categorized as TN.

Table 4.2: Definitions of *TP*, *FP*, *TN*, and *FN*.

	Building	Background
Classified as Building	*TP*	*FP*
Classified as Background	*FN*	*TN*

After each segment is classified, performances of the algorithms are evaluated by using the well-known three metrics [158] which are defined as

$$\text{precision} = \frac{TP}{(TP+FP)},\ \text{recall} = \frac{TP}{(TP+FN)},\ \text{f-score} = 2\frac{\text{precision.recall}}{(\text{precision}+\text{recall})}.$$

It is important to remember that the precision metric disregards the missed buildings. On the other hand, the recall pay no attention on false alarms. However, in order to evaluate the building detection performance, it is crucial to consider both false alarms and missed buildings. For that reason f-score, which measures the harmonic mean of precision and recall, is used in the experiments.

In Mean Shift segmentation, *minArea* is selected as 300 where the minimum building size in our test site is approximately $120m^2 \approx 400$ pixel. In order to select the optimum h_s and h_r parameters, the training images are segmented by different $h_s \in \{4,5,6,7,8\}$ and $h_r \in \{4,5,6,7,8\}$ parameters. Then, the Overall Segmentation Quality(OSQ) is calculated using these segmentation results. The set of optimal parameters, which gives the highest OSQ, is used in the segmentation of the test images. After the segmented images are pre-processed, the features given in Section 4.1.1.3 are extracted from the segments.

For each experiment, the training set is divided into two subsets. By using the first set, FSG is trained with different k values (5, 10, 15 and 20) at the base-layer, and the rest of the training set is used to validate the performance of the meta-layer classifier. The k value which gives the highest f-score at the meta-layer of FSG in the training phase is selected, and this k value is used for fuzzy k-nn classifiers at the base-layer during the test phase.

After FSG is trained, the features of the test images are fed to the base-layer classifiers. Decisions of the base-layer classifiers are fused at the meta-layer to construct a meta-layer fusion space, and a meta-layer classifier is employed in the fusion space to detect buildings.

Moreover, performance of FSG is compared with SVM (*SVM-Linear* and *SVM-RBF*), and fuzzy k-nn classifiers, which are employed on an aggregated feature space $F = F_1 \times \ldots \times F_j \times \ldots \times F_J$. Finally, FSG is compared with one of the recent approaches proposed by Turker and Koc San [138] without including normalized Digital Surface Model, since the dataset does not

(a) Locations of Patches in Ankara (b) Image Sample 1 (c) Image Sample 2

(d) Image Sample 3 (e) Image Sample 4 (f) Image Sample 5

(g) Image Sample 6 (h) Image Sample 7 (i) Image Sample 8

Figure 4.2: The example images from Ankara dataset used in the experiments.

contain stereo images. Therefore, the set of features employed in *Data-Set 13* (DS13) which is proposed by Turker and Koc San [138] is used. During these experiments, SVM parameters are selected using cross-validation as suggested in [159].

The individual performances of the base-layer fuzzy k-nn classifiers, that averages over 10 experiments, are given in Table 4.3. Two sets of experiments are accomplished to compare the classification performances of the suggested decision fusion method, and the classical single classifier approach.

In the first set of experiments, base-layer classifiers of the suggested FSG architecture is fed by a set of 15 different features given in Table 4.1, which is called Feature Set-1.

As it can be seen from Table 4.4, the decision fusion approach improves the highest performance obtained at the output of the fuzzy k-NN classifier. Then, the same set of features are concatenated and the 128-dimensional feature vectors are fed to a single SVM and a single fuzzy k-NN classifier. The performances of SVM and fuzzy k-nn get worse than the performances of the classifiers fed by only histogram mean value feature. Note that the suggested decision fusion method outperforms all the classifier methods, given in Table 4.4.

Decisions of base-layer classifiers and FSG are visualized in Figure 4.3 for the first sample, Sample 1 in the dataset. In the label masks,

- yellow pixels belong to the set of pixels in TP,
- red pixels belong to the set of pixels in FP,

Table 4.3: Performance results of the individual base-layer fuzzy k-nn classifiers using individual features.

	precision	recall	f-score
stdc	71.31%	67.17%	69.18%
hist_pdv	86.78%	77.28%	81.75%
hist_m	85.92%	79.61%	**82.64**%
hist_v	60.74%	61.04%	60.89%
hist_s	81.80%	76.48%	79.05%
hist_k	64.44%	67.18%	65.78%
hist_energy	64.32%	67.92%	66.07%
hist_entropy	68.26%	52.63%	59.43%
rectangularity	67.38%	67.97%	67.67%
axis_lengths	66.19%	67.44%	66.81%
area	59.22%	57.90%	58.55%
glcm_contrast	68.58%	75.62%	71.93%
glcm_correlation	64.77%	74.62%	69.35%
glcm_energy	67.39%	78.93%	72.71%
glcm_entropy	71.10%	80.45%	75.49%

Table 4.4: Performance results of FSG with other classification approaches using all of the features.

	precision	recall	fscore
FSG	88.86%	78.67%	83.46%
SVM-Linear	78.84%	78.00%	78.42%
SVM-RBF	58.90%	83.48%	69.07%
Turker DS13-RBF	58.13%	62.37%	60.18%
Turker DS13-Linear	90.48%	76.58%	82.95%

- green pixels belong to the set of pixels in FN, and
- blue pixels belong to the set of pixels in segment boundaries.

In Table 4.4, the recall of the classifier employed on *hist_mean* is greater than the recall of the FSG, and its precison is less than that of the FSG. However, the f-score of the classifier employed on *hist_mean* is better than that of the other base-layer classifiers, and is close to the f-score of the FSG.

One of the main reasons of this performance relationship between the base-layer classifier employed on *hist_m* and the FSG is the color dominance over the other features in the dataset. For instance, let us denote the sets of segments which contain buildings with red roofs as $\{re_i\}_{i=1}^{N}{}_r$ and buildings with white roofs as $\{wh_i\}_{i=1}^{N}{}_w$.

Since $N_r > N_w$ in the dataset, color features of the segments in $\{re_i\}_{i=1}^{N}{}_r$ may construct clusters in the color feature spaces. Then, color features of the segments in $\{wh_i\}_{i=1}^{N}{}_w$ may reside far from the clusters. Therefore, average performances of base-layer classifiers employed on the color feature spaces, such as *hist_m* and *hist_pdv*, are greater than the performances of the other base-layer classifiers.

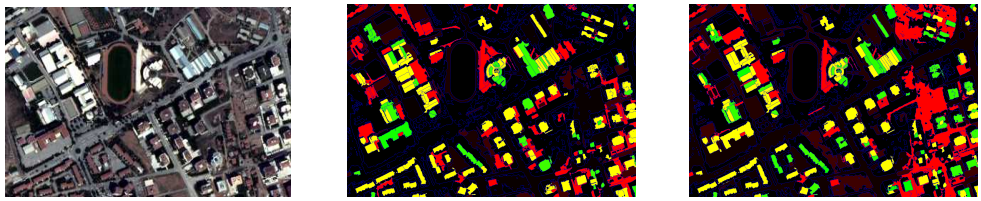
If dense clusterings occur in color feature spaces, then base-layer classifiers employed on color feature spaces decide the class of the segments in $\{wh_i\}_{i=1}^{N}{}_w$ as background with large class posterior probability in decision spaces. If the segments in $\{wh_i\}_{i=1}^{N}{}_w$ have discriminating shape and textural characteristics, then class posterior probabilities of the segments belonging to $\{wh_i\}_{i=1}^{N}{}_w$ may decrease when the decisions of the base-layer classifiers employed on color feature spaces are fused with that of the other base-layer classifiers in fusion space. Therefore, the FSG provides higher precision values than the base-layer classifiers.

This case can be observed by investigating the results given in Figure 4.3.p and Figure 4.3.q. In the upper left part of Figure 4.3.p, a segment which contains pixels depicted with green color is classified as background although its ground truth is building. On the other hand, that segment is correctly labeled by the other 10 base-layer classifiers employed on shape and texture features. Therefore, FSG correctly predicts its label by decision fusion in Figure 4.3.q.

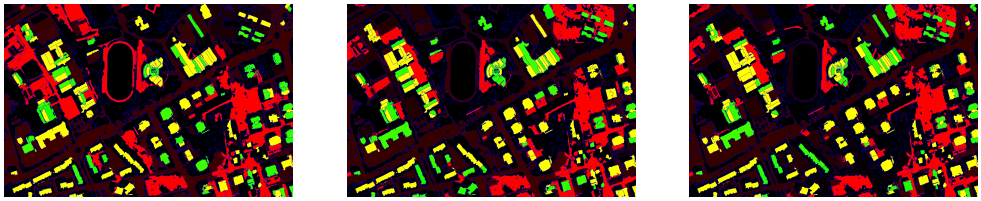
If sparse clusterings occur in color feature spaces, then base-layer classifiers employed on color feature spaces decide the class of the segments in $\{re_i\}_{i=1}^{N}{}_r$ as building with small class posterior probabilities in decision spaces. If these segments have the similar shape and textural properties with the other objects such as roads and farms, then base-layer classifiers that use shape and texture features may predict labels of these segments either correctly or incorrectly with large class posterior probability. Then, these segments may have large decision class posterior probabilities in fusion space. Therefore, FSG may misclassify these segments (see Figure 4.3).

In the second set of the experiments, the robustness of the FSG is examined with respect to the random feature selection for building detection. For this purpose, a subset of the 15 features, called Feature Set-2 is selected, and the performance of FSG method is compared with that of SVMs and base-layer classifiers.

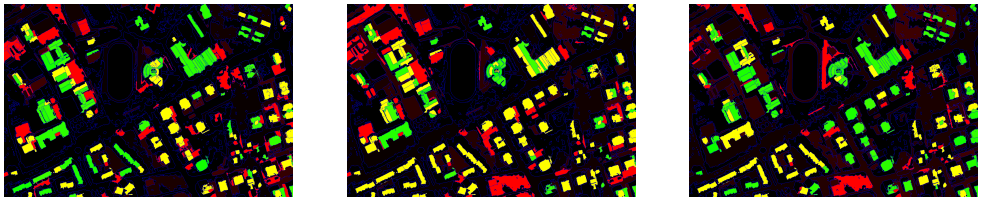
An example of these tests for 7 randomly selected features (*stdc*, *hist_v*, *hist_k*, *hist_entropy*, *area*, *axis_lengths* and *rectangularity*) is given in Table 4.5. The results for Feature Set-1 and Feature Set-2 are given in Figure 4.4 and 4.5, respectively.



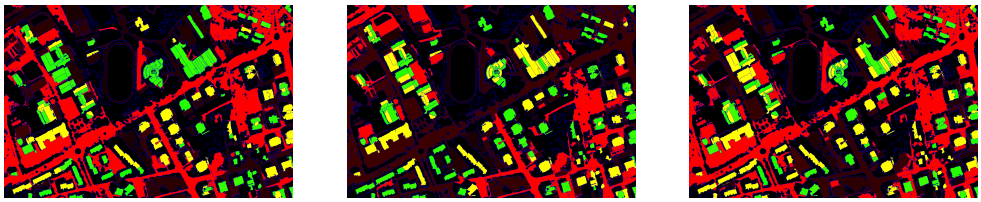
(a) RGB image (b) *area* (c) *glcm_contrast*



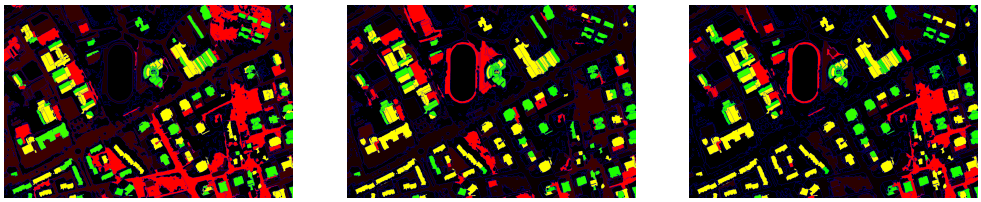
(d) *glcm_correlation* (e) *glcm_energy* (f) *glcm_entropy*



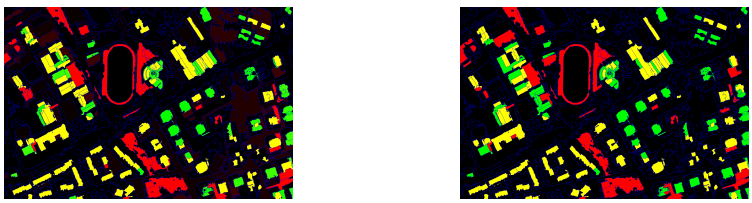
(g) *axis_lengths* (h) *rectangularity* (i) *stdc*



(j) *hist_entropy* (k) *hist_v* (l) *hist_k*

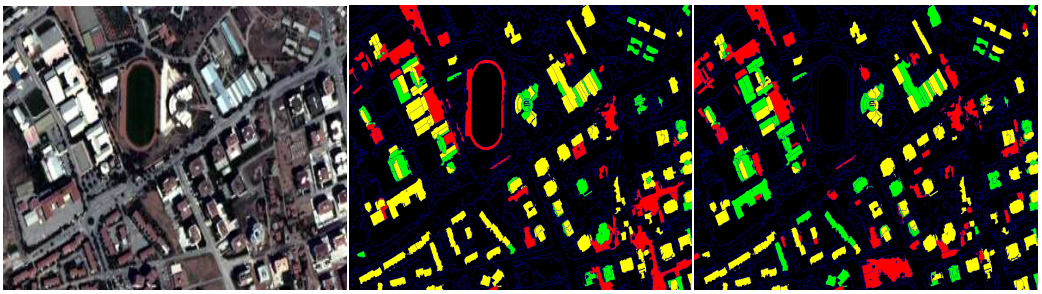


(m) *hist_energy* (n) *hist_s* (o) *hist_pdv*

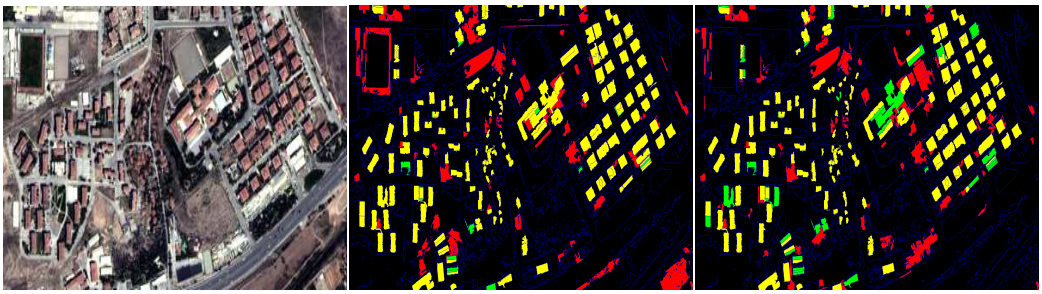


(p) *hist_m* (q) FSG

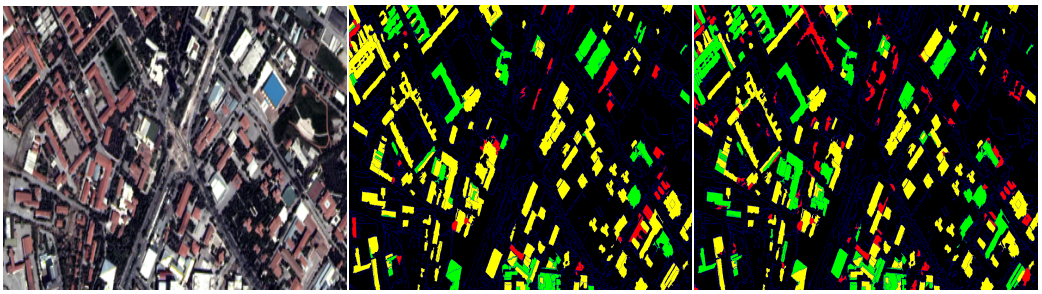
Figure 4.3: Building Detection with Decision Fusion in an experiment on Image Sample 1 using Feature Set-1. Decisions of the base-layer classifiers on the feature spaces are given in (b)-(p) and the decision of FSG is given in (q).



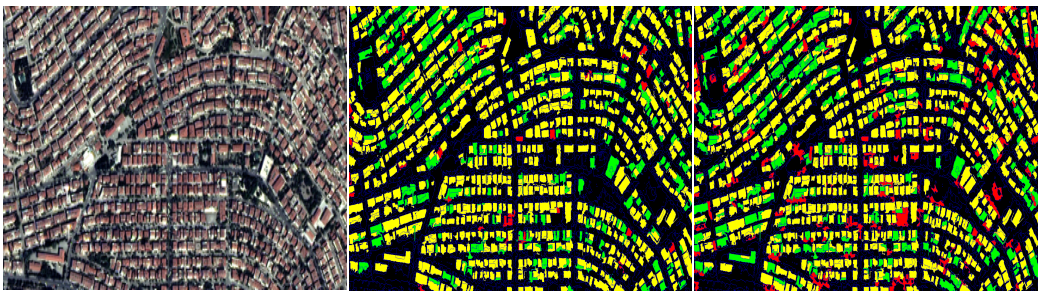
(a) Image Sample 1 (b) Results on Feature Set-1 (c) Results on Feature Set-2



(d) Image Sample 2 (e) Results on Feature Set-1 (f) Results on Feature Set-2

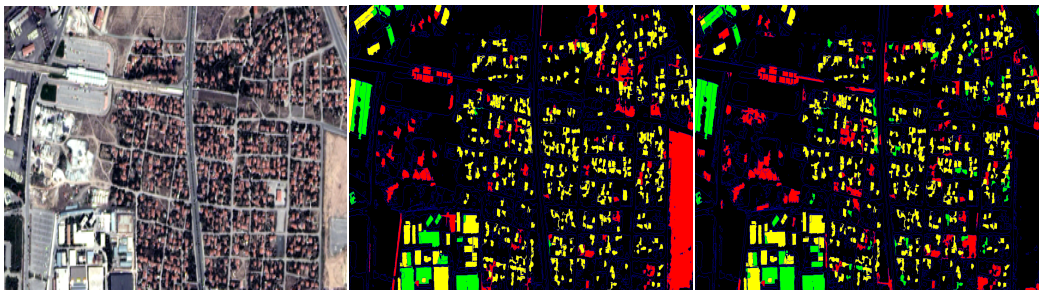


(g) Image Sample 3 (h) Results on Feature Set-1 (i) Results on Feature Set-2

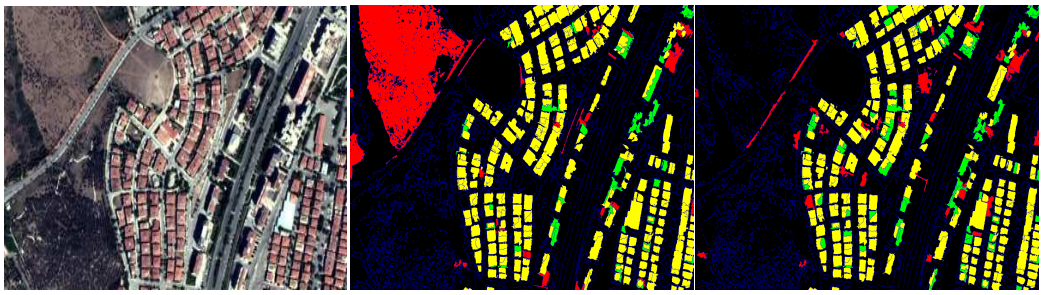


(j) Image Sample 4 (k) Results on Feature Set-1 (l) Results on Feature Set-2

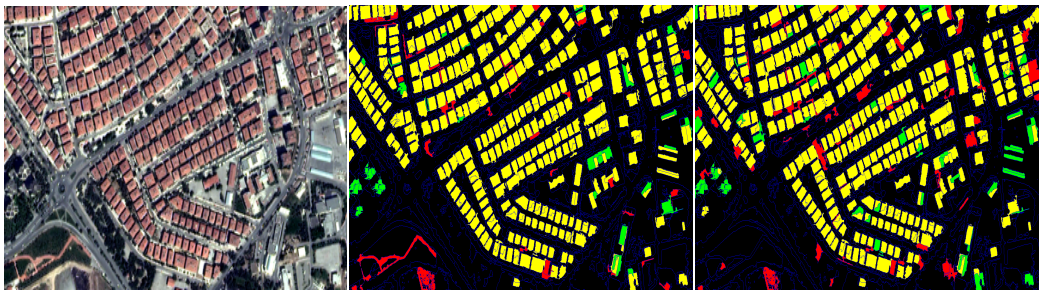
Figure 4.4: Results of FSG on the sample images 1-4, where $k = 10$ is selected for base-layer fuzzy k-nn classfiers.



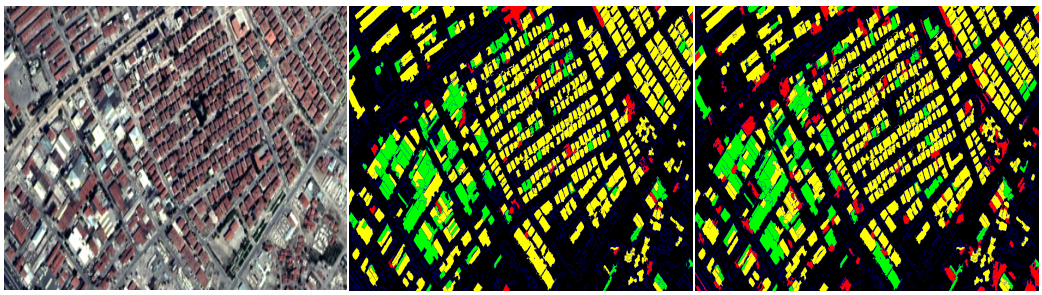
(a) Image Sample 5 (b) Results on Feature Set-1 (c) Results on Feature Set-2



(d) Image Sample 6 (e) Results on Feature Set-1 (f) Results on Feature Set-2



(g) Image Sample 7 (h) Results on Feature Set-1 (i) Results on Feature Set-2



(j) Image Sample 8 (k) Results on Feature Set-1 (l) Results on Feature Set-2

Figure 4.5: Results of FSG on the sample images 5-8, where $k = 10$ is selected for base-layer fuzzy k-nn classfiers.

In all of the tests, the FSG outperforms the f-scores of the individual classifiers at the base-layer. However, f-score of the SVM with RBF Kernel is 0.44% higher than that of the FSG. In addition, the f-score of the FSG in Table 4.5 is less than the f-score of the FSG in Table 4.4. Therefore, feature space selection is critical for the FSG.

Table 4.5: Performance results of the experiments on Feature Set-2.

	precision	*recall*	*fscore*
stdc	71,31%	67,17%	**69,18%**
hist_v	60,74%	61,04%	60,89%
hist_k	64,44%	67,18%	65,78%
hist_entropy	68,26%	52,63%	59,43%
area	59,22%	57,90%	58,55%
axis_lengths	66,19%	67,44%	66,81%
rectangularity	67,38%	67,97%	67,67%
FSG	74,27%	69,83%	71,98%
SVM-Linear	61,29%	55,83%	58,43%
SVM-RBF	56,78%	99,95%	72,42%

4.2 Target Detection for Multi-sensor Decision Fusion

Sensors with multiple modalities have the capability of sensing the environment by evaluating the data which represent the different characteristics of the environment. Therefore, the manipulation and the integration of different type of sensors by Data Fusion algorithms is an important obstacle for robotics research. One of the challenging problems of Multi-sensor Data Fusion is to select the *best* information extractors that manipulate on the multimodal data which are obtained from different sensors, and achieve inference from the data by reducing the sensor inaccuracy and the environment uncertainty, thereby, the entropy of the data representing the environment. Since the physical modality of each group of data obtained from each individual sensor is discrete and divergent, individual information extractors which are expert on each modality are required. Moreover, the information extractors should be complementary in order to supply the decisions made by each individual expert.

Decision Fusion algorithms, which employ ensemble learning approach such as Adaboost, often process the data sampled from the same distribution and they are experienced with overfitting of the data. Therefore, most of the Decision Fusion systems may not satisfy the requirements of multimodal sensor fusion such as the manipulation of the heterogeneous data by considering the generalization error or risk minimization criteria. In order to meet these requirements, FSG is implemented for the object detection in this section. The object detection problem is considered as a multi-class classification problem, where background and each object of interest belong to individual classes.

4.2.1 Problem Definition and Scenario for Data Acquisition

In the multi-modal target detection problem, data acquisition is accomplished by an audio-visual sensor, which is a webcam with a microphone located in an indoor environment as shown in Figure 4.6. In this scenario, recordings of the audio and video data are obtained from randomly moving two targets T_1 and T_2, i.e. two randomly walking people, in the indoor environment. The problem is defined as the classification of the audio and video frames with two targets in the noisy environment, where the other people talking in the environment and the obstacles distributed in the room are the sources of the noise for audio and video data.

Four classes are defined for the dataset. The first class represents the absence of the moving targets, in other words, there is no target in the environment. The second and the third classes represent the existence of the first and the second target in the environment. In the fourth class, both of the targets take place in the environment. Definitions of the classes according to the presence and absence of two targets T_1 and T_2 in the environment are given in Table 4.6.

Table 4.6: Definitions of classes, according to the presence (★) and absence (○) of two targets, T_1 and T_2.

	Class1	Class2	Class3	Class4
T_1	○	★	○	★
T_2	○	○	★	★

The audio characteristics of the targets are determined with specific musical melodies with

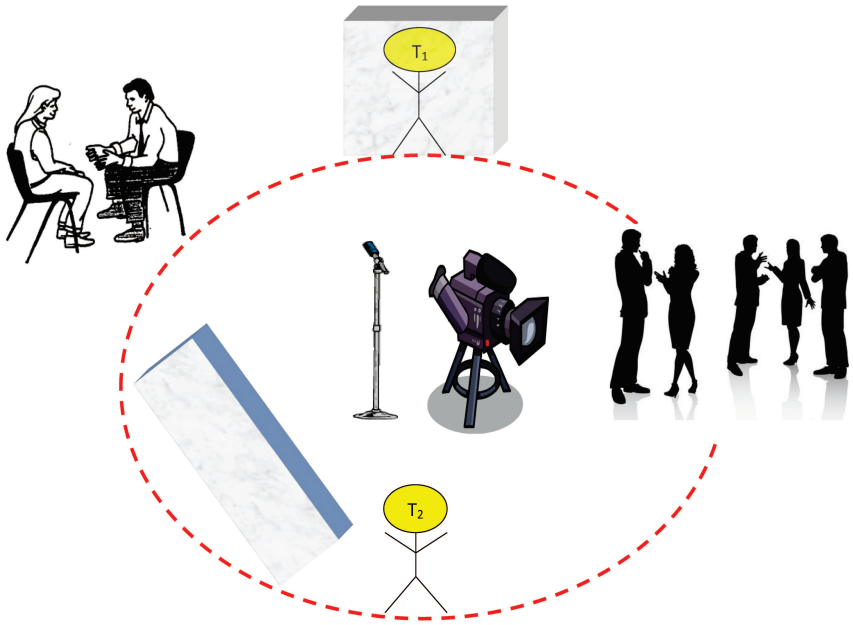


Figure 4.6: The data acquisition setup for the multi-modal decision fusion.

Table 4.7: Number of samples.

	Class1	Class2	Class3	Class4	Total
Train	190	190	190	189	759
Test	190	190	160	189	729

different tonalities. In Table 4.7, the number of samples (image frames) belonging to each class for each data set is given.

The experimental setup is designed to satisfy the one of the requirements of FSG, which is the correct classification of the samples by at least one of the base-layer classifiers, for the performance boost given in Chapter 3. Therefore, complementary expertise of the base-layer classifiers on different classes is aimed to achieve.

For instance, if a target is hidden behind an obstacle such as a curtain (see Figure 4.7), then a base-layer classifier which employs audio features for classification can correctly detect the target behind the curtain, even if a a base-layer classifier which employs visual features for classification, cannot detect the target correctly.

4.2.1.1 Audio-Visual Descriptors for Multi-modal Target Detection

In order to extract visual features from the frames of a video recording, two MPEG-7 descriptors, Homogenous Texture (HT) and Color Layout (CL) described in the previous chapter are used. Three audio descriptors, Fluctuation (Fluct.), Chromagram (Chrom.) and Mel-

Figure 4.7: A sample frame used in the training dataset in which a target (T_1) is hidden behind an obstacle which is a curtain.

Frequency Cepstral Coefficients (MFCC), [160] are used to extract audio features. Audio descriptors are described briefly below.

- **Mel-Frequency Cepstral Coefficients** (Feature Space Dimension $D_j = 25$) describes the spectral shape of an audio signal. In the algorithm, first the Fourier transform is employed on the audio signal, and the powers of the spectrum are mapped onto the Mel-scale. Then, a Discrete Cosine Transform(DCT) is applied to the logarithms of the powers of Mel-scale frequencies. Finally, the amplitudes of the DCT spectrum are considered as MFCC features.

- **Fluctuation** (Feature Space Dimension $D_j = 35$) estimates the rhythmic periodicity in the audio signal. Similar to MFCC, this is accomplished by the spectral analysis of the signal. First the spectrogram is computed on audio frames and the Terhardt outer ear modeling is computed, with Bark-band redistribution of the energy, and estimation of masking effects [160]. Then, amplitudes of the Fast Fourier Transform (FFT) spectrum computed on each Bark band are used as features.

- **Chromagram** (Feature Space Dimension $D_j = 12$) is used to measure the tonality in an audio signal by computing the energy distribution of pitches. Briefly, frequency components of the audio are computed and converted to a spectrogram using the FFT. Spectrogram is filtered by selecting the frequencies in the range of $[100Hz, 5000Hz]$ and the local maximum values of the spectrum are selected. Finally, variables of a feature vector, called Chroma, are computed by estimating the energy distribution along 12 pitch classes [160].

4.2.2 Experiments

In the experiments, three different fusion approaches are considered; FSG for the fusion of the decisions of the classifiers employed on i) visual features (***Video Fusion***), ii) audio features

(***Audio Fusion***) and *iii*) both audio and visual features (***Audio-Visual Fusion***).

Experimental results show that the base-layer classifiers employed on visual features perform better than the classifiers employed on audio features for the fourth class. However, the classifiers employed on audio features perform better than the classifiers employed on visual features for the first three classes. For instance, the base-layer classifiers employed on the visual descriptors most likely misclassify the samples from the second class, but perform better than the other classifiers for the fourth class (see Table 4.8 and Table 4.8). On the other hand, the base-layer classifiers employed on audio descriptors have a better discriminative power compared to the base-layer classifiers employed on the visual descriptors for the first class.

One of the reasons of this observation is that the classifiers employed audio features which are affected by audio noise, are less sensitive to *noise* than the classifiers employed on visual features which are affected by visual noise. In other words, two targets have visual appearance properties similar to the other objects in the environment and the obstacles, such as curtains and doors, block completely the visual appearance of the targets. On the other hand, the targets have different visual appearance properties such that the heights of the targets and color of their clothes are different from each other. In addition, the audio properties of the measurements obtained from the targets have discriminative characteristics which are different than the other objects in the environment.

An analysis of Table 4.8 and Table 4.9 reveals that the performance of an individual descriptor varies across the classes due to similar arguments. As a result, a substantial increase in the general classification performance of the FSG is achieved.

Table 4.8: Classification performances for training dataset.

	Class1	Class2	Class3	Class4	Total
Homogeneous Texture	76.84%	67.89%	76.84%	96.30%	79.45%
Color Layout	93.16%	86.84%	84.21%	**97.35%**	90.38%
MFCC	**99.47%**	84.74%	**94.74%**	83.60%	**90.65%**
Chromagram	98.42%	**90.00%**	89.47%	82.01%	89.99%
Fluctuation	94.74%	85.79%	75.79%	52.38%	77.21%
Video Fusion	92.63%	87.37%	84.21%	95.77%	89.99%
Audio Fusion	97.89%	93.16%	96.32%	92.59%	94.99%
Audio-Visual Fusion	**99.47%**	**97.89%**	98.42%	**100%**	**98.95%**

Table 4.9: Classification performances for test dataset.

	Class1	Class2	Class3	Class4	Total
Homogeneous Texture	54.74%	49.47%	43.75%	**93.12%**	60.91%
Color Layout	76.32%	49.47%	40.63%	83.07%	63.24%
MFCC	92.11%	77.37%	**93.13%**	81.48%	**85,73%**
Chromagram	92.63%	**84.21%**	83.13%	66.67%	81.62%
Fluctuation	**93.68%**	82.63%	75.00%	52.38%	75.99%
Video Fusion	69.47%	54.21%	45.63%	90.48%	65.71%
Audio Fusion	90.53%	93.16%	93.13%	79.37%	88.89%
Audio-Visual Fusion	93.68%	**94.21%**	94.37%	97.88%	95.06%

Each cell of Table 4.10 and Table 4.11 represents the number of samples which are misclassified by the classifier for the descriptor in the i^{th} row, and correctly classified by the classifier for the descriptor in the j^{th} column, for the training and test datasets, respectively. In the tables, the maximum number of misclassified samples for each descriptor is bolded.

For example, 144 samples which are misclassified in HT feature space are correctly classified in Chromagram feature space. The samples that are misclassified in the feature spaces defined by the visual descriptors are correctly classified in the feature spaces defined by the audio descriptors. This is observed when the visual appearance of the targets are affected by the visual noise, e.g. the targets are completely blocked by an obstacle, such as a curtain, but their sounds are clearly recorded by the audio sensor, as shown in Figure 4.7. Therefore, it can be easily observed from the tables that the feature spaces are complementary to each other.

On the other hand, the samples that are misclassified in the feature spaces defined by the audio descriptors (e.g. Fluctuation and Chromagram) are correctly classified in the feature spaces defined by the visual descriptors (e.g. CL and HT) when there are other objects that make sounds with audio characteristics similar to the targets in the environment. In this case, audio features of the targets are affected by audio noise. If the visual sensor can make *clear* measurements on the targets, such that the visual features are not affected by visual noise, then the classifiers employed in the feature spaces defined by the visual descriptors can correctly classify the samples.

Table 4.10: Covariance matrix for the number of correctly and misclassified samples for the descriptors for train dataset.

Train Dataset		**Correct Classification**					
		HT	CL	MFCC	Chrom.	Fluct.	Total
Misclassification	HT	0	137	142	**144**	130	156
	CL	54	0	**64**	59	57	73
	MFCC	57	**62**	0	44	40	71
	Chromagram	**64**	62	49	0	39	76
	Fluctuation	147	**157**	142	136	0	173

Table 4.11: Covariance matrix for the number of correctly and misclassified samples for the descriptors for test dataset.

Test Dataset		**Correct Classification**					
		HT	CL	MFCC	Chrom.	Fluct.	Total
Misclassification	HT	0	134	247	**249**	233	285
	CL	117	0	**235**	223	216	268
	MFCC	66	**71**	0	52	54	104
	Chromagram	**98**	89	82	0	61	134
	Fluctuation	**123**	**123**	**125**	102	0	175

4.2.2.1 Statistical Analysis of Feature, Decision and Fusion Spaces

In this section, transformations of class conditional distributions of feature and decision vectors through the layers of the FSG in feature, decision and fusion spaces are analyzed for target detection problem described in the previous section. Histograms are used to approximate the distributions for visualization.

In the histogram representation [161], first the range of random variables A that reside in $[0,1]$ (e.g., posterior probabilities computed by base-layer classifiers in the FSG), is divided into B intervals (low_b, up_b), $b = 1,2,\ldots,B$ representing the width $w_b = up_b - low_b$ of the b^{th} *bin* of a histogram. Denoting the probability density function of A as f, the entropy can be approximated using

$$\mathbb{H}(A) \approx -\sum_{b=1}^{B} p_b,$$

where the probability of a bin

$$p_b = \int_{low_b}^{up_b} f(a)da, \ \forall b = 1,2,\ldots,B$$

is approximated as the area of a rectangle of height $f(a_b)$ which is $w_b f(a_b)$ where a_b is a representative value within the interval (low_b, up_b). Then the entropy is approximated as

$$\mathbb{H}(A) \approx -\sum_{b=1}^{B} p_b \log \frac{p_b}{w_b}.$$

In order to represent the class conditional distributions using histograms, first the number of samples belonging to each class is computed by counting $n_{b,c}$ which is the number of samples that fall into each of the disjoint histogram bins $b = 1,2,\ldots,B$ for each class $c = 1,2,\ldots,C$, as

$$N_c = \sum_{b=1}^{B} n_{b,c}$$

and $N = \sum_{c=1}^{C} N_c$ where N is the number of samples in the dataset. Then a bin probability is computed as $p_{b,c} = \frac{n_{b,c}}{N}$.

Note that, three different spaces are constructed through the FSG; *i*) feature space (at the input of base-layer classifiers), *ii*) decision space (at the output of base-layer classifiers) and *iii*) fusion space (at the input of meta-layer classifier). Feature spaces consist of the feature sets obtained from the descriptors. A bin probability $p_{b,c}$ of a histogram computed in a feature space F_j is an approximation to a class conditional distribution. In a decision space of a classifier employed on F_j, posterior probability or class membership vectors $\bar{\mu}(\bar{x}_{ij})$ are used. In the fusion space, the histograms are computed using the concatenated membership vectors $\bar{\mu}(\bar{x}_i)$.

In Figure 4.8, the histograms representing the approximate probability distributions at each base-layer decision space (Figure 4.8 (a-e)) and the fusion space (i.e. the meta-layer input feature space) (Figure 4.8.f) are displayed for test dataset. It is observed from the histograms that the concatenation operation decreases the uncertainty of the feature spaces.

Entropy values computed in feature spaces are given in Table 4.12. Entropy values computed for each class in each feature space give the information about the data uncertainity in the

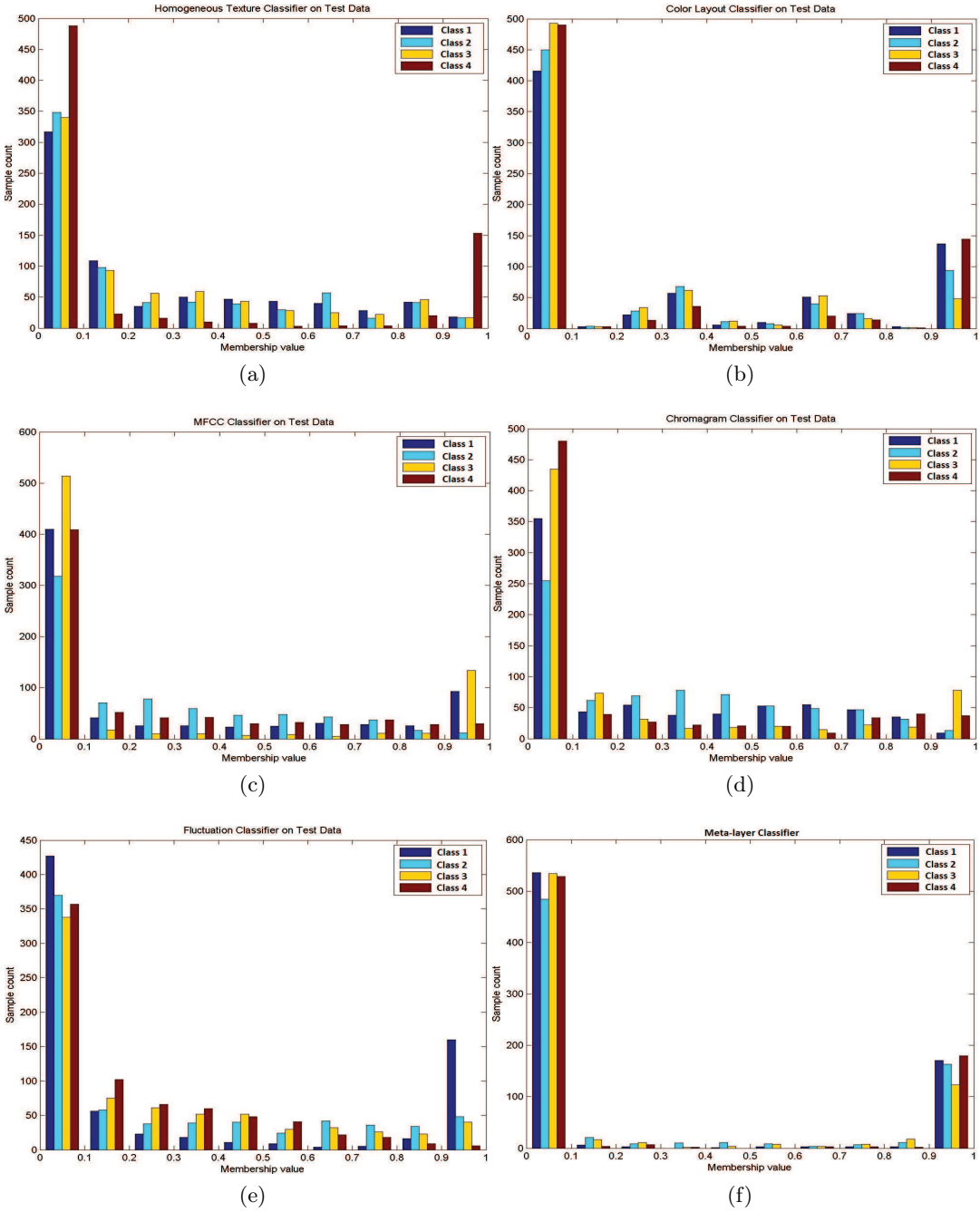


Figure 4.8: Histograms which represent distributions for the individual decision spaces of base-layer classifiers employed using (a) Histogram Texture, (b) Color Layout, (c) MFCC, (d) Chromagram, (e) Fluctuation features, and (f) in the fusion space of the meta-classifier in FSG. Notice that the lowest entropy is observed in the fusion space.

feature space. If the distributions of the features in a feature space F_j provide lower entropy values for a particular class ω_c than the other classes, then the features may represent a characteristic of class ω_c. Therefore, a classifier employed on F_j classify the samples belonging to ω_c with better performance than the samples belonging to other classes.

For instance, distributions of Fluctuation, MFCC and Homogeneous Texture features have the lowest entropy values for the first, the third and the fourth classes, respectively (see Table 4.12). The base-layer classifiers which use these features provide the highest classification performances, as shown in Table 4.9.

Although the distribution of Color Layout features gives the lowest entropy for the second class than the other audio and visual features, a base-layer classifier employed on Color Layout features performs worse than the other classifiers employed on other features. However, the features of the samples belonging to the fourth class have the lowest entropy in Color Layout feature space (see the row of Table 4.12 labeled Color Layout). Then, a classifier employed on Color Layout feature space gives the highest classification performance for the fourth class as given in Table 4.9.

Entropy values computed in decision and fusion spaces are given in Table 4.13 for test dataset. Entropy values of the class membership vectors in decision spaces represent the decision uncertainty of base-layer classifiers for each class. Note that the classifiers employed on the feature spaces with minimum decision uncertainties for particular classes provide the highest classification performances for these classes (see Table 4.9).

Entropy values of the membership vectors $\bar{\mu}(\bar{x}_i)$ that reside in the fusion space represent the joint entropy of $\{\bar{\mu}(\bar{x}_{i,j})\}_{j=1}^{J}$, since $\bar{\mu}(\bar{x}_i) = [\bar{\mu}(\bar{x}_{i,1}) \dots \bar{\mu}(\bar{x}_{i,j}) \dots \bar{\mu}(\bar{x}_{i,J})]$. If classifier decisions are independent, then the entropy value Ent_{fusion} of $\bar{\mu}(\bar{x}_i)$ is equal to the sum of the entropy values Ent_j of $\bar{\mu}(\bar{x}_{i,j})$, $\forall j = 1, 2, \dots, J$, such that

$$Ent_{fusion} = \sum_{j=1}^{J} Ent_j.$$

However, $Ent_{fusion} \leq \sum_{j=1}^{J} Ent_j$ in Table 4.13, which implies that the decisions are dependent. This dependency occurs by sharing the samples among the classifiers in the FSG as shown in Table 4.11. Thereby, lower entropy values are obtained in the fusion space.

Table 4.12: Entropy values computed in feature spaces for test dataset.

Feature Spaces	Class 1	Class 2	Class 3	Class 4
Homogeneous Texture	0.3751	0.3840	0.3702	**0.0679**
Color Layout	0.1905	**0.2644**	0.3255	0.0861
MFCC	0.1920	0.3824	**0.0879**	0.3347
Chromagram	0.3442	0.3621	0.2011	0.2834
Fluctuation	**0.0389**	0.3013	0.3115	0.4276

Table 4.13: Entropy values computed in decision and fusion spaces for test dataset.

Decision and Fusion Spaces	Class 1	Class 2	Class 3	Class 4
Homogeneous Texture	0.2160	0.2360	0.2550	**0.0457**
Color Layout	0.1057	0.3052	0.2383	0.4584
MFCC	0.1539	0.2161	**0.1322**	0.1936
Chromagram	0.1165	**0.1092**	0.1582	0.1760
Fluctuation	**0.0344**	0.2286	0.2890	0.3228
Fusion Space	*0.0228*	*0.0529*	*0.0873*	*0.0156*

4.3 Chapter Summary

In this chapter, the proposed FSG algorithm is applied to target detection problems in remote sensing and audio-visual data processing applications. First a new building detection technique is proposed using decision fusion with FSG. Note that the dimension of the concatenated feature vectors employed at the base-layer classifiers is quite high (i.e. a 128 dimensional vector) in building detection problem. Decomposing this vector into the base-layer classifiers according to their modalities reduces the curse of dimensionality problem.

Fusion space created by the Cartesian product of decision spaces also has some nice properties. First of all, since the membership vectors lie in the interval of $[0, 1]$, there is no need for normalization of the feature vectors. Secondly, the dimension of the fusion space is fixed to $2\times$ number of base-layer classifiers, which is independent of the dimension of the feature vectors fed to base-layer classifiers. Therefore, one may include various types and sizes of the feature vectors for a fixed number of base-layer classifiers without increasing the size of the decision space.

In building detection problem, it is observed that an FSG with various parameter and feature sets mostly improves the f-score performance of the individual base-layer classifiers. In addition, fusion of decisions of base-layer classifiers employed on different feature sets enables us learning different characteristics of the segments which represent the buildings. In this technique, it is possible to optimize the individual parameter sets of the base-layer classifiers, independently, using FSG.

One of the crucial points of the proposed fusion technique is the design of the feature spaces of the individual base-layer classifiers. One should pay attention to select *complimentary* features of the buildings to assure the coverage of the characteristics of the various types of segments. For instance, a building can be detected in difficult scenes where the building has similar color to the background, and the other buildings have different textural or geometric properties than the background. In this case, color features may not provide useful information to the base-layer classifiers for the detection, but the classifiers can extract discriminative information from the texture and shape features. Then, the base-layer classifiers can be trained to learn the discriminative properties of the features, and provide their decisions as the feature vectors to the meta-layer classifiers, which can correctly detect the buildings. Recall that, detailed discussion on the sufficient and necessary conditions required for feature extraction and sample selection in FSG is given in Chapter 3.

Similar to the most of the other supervised building detection algorithms [81, 69, 138], training and test images are selected from the same region. This approach assumes the statistical

stability of the training and test data. If the training data and test data are selected from dissimilar and irrelevant regions (where data and domain shift occurs), the performance values decrease drastically, due to the change in the terrain structures and building variations. In practice, this challenge occurs if the buildings in training and test images have different sizes and color/shading properties. Similar problems are addressed in various remote sensing applications such as land use classification in [23, 25]. In order to solve this problem, state-of-the-art domain adaptation methods [22, 23, 24, 25] can be used.

Additionally, the proposed approach requires adequate and reliable ground-truth labels to obtain the decisions of the base-layer classifiers. In order to relax this requirement, semi-supervised and unsupervised classification methods can be employed in the base-layer classifier by satisfying the requirements proposed in Chapter 3.

In the second application, the data is obtained from different sensors, namely audio and video recorders. This multi-modal data is fused under the FSG architecture. Apparently, the features extracted from the individual modes have different statistical properties, and give diverse information about different classes. Therefore, base-layer classifiers each of which can correctly classify the samples belonging to specific classes can be trained, even if the individual performances of the classifiers are low. Since this data setting is complementary to the observations on the synthetic datasets (see Section 3.7.1) and satisfy the requirements mentioned in Chapter 3, the FSG boosts the performances of the base-layer classifiers with 10% performance gain.

Moreover, it is observed that the entropies of the features are decreased through the feature space transformations from the base-layer to the meta-layer of the architecture. Therefore, the architecture transforms the linearly non-separable feature spaces with higher dimensions into a more separable feature space (Fusion Space) with lower dimensions which are fixed with the number of classes and base-layer classifiers.

Part II

Decision Fusion for Unsupervised and Semi-supervised Learning

CHAPTER 5

FUSION OF MULTIPLE IMAGE SEGMENTATION ALGORITHMS

Image segmentation is one of the most important, yet unsolved problems in computer vision and image processing areas [162, 163, 164]. Various segmentation algorithms in the literature have been applied to segment the objects in images. However, there are two main issues for their employment.

The first challenge is to *extract* a robust structure, e.g. shape, of the segments by analyzing the outputs of segmentation algorithms when a target segmentation is not available with a training dataset. This challenge has been studied in data mining community as a consensus clustering problem [165]. In this work, the problem has been considered as an unsupervised *image mining* problem and analyzed using a consensus segmentation method called Segmentation Fusion.

The second challenge is the selection of an *appropriate* algorithm or its parameters that provides an *optimal* segmentation which is closer to a *target* segmentation if a target segmentation is available with a training dataset. This challenge has been analyzed by a new segmentation fusion algorithm called Semi-supervised Segmentation Fusion which incorporates prior and side information obtained from training datasets and expert knowledge to the proposed Segmentation Fusion algorithm.

In real-world applications, the problem definition of segmentation varies by the type of data and the targets to be detected. For instance, algorithms based on density estimation methods may be successfully employed to detect targets that are distributed in the image. However, these algorithms may fail if the global statistics of the data are not smooth or predictable. Moreover, a global distribution function that defines the global statistics of the data may not be available. In these problems, structural or local segmentation algorithms should be employed. The choice of a suitable segmentation algorithm in many problems requires computationally exhaustive analyses in the datasets. Even with the choice of a single algorithm to employ for a specific problem, different algorithm parameters results in different segmentations. Then, the selection of an *optimal* segmentation becomes an extremely difficult task.

A partial solution to the above mentioned problems is to train the parameters of a segmentation algorithm which is selected by heuristics. This is a challenging problem for specific applications, such as the segmentation of remotely sensed images, since the labels of the data may not be available or the size of the data may be either too large or not sufficient for training. Moreover, the images in training and test datasets may not be statistically stable. In addition, in most practical problems, there is no *best* set of parameters to be selected for a segmentation algorithm. This is basically because of the inherently multi-resolution nature

of the problem. In other words, some objects are recognized on large scales whereas other objects may require high resolution data. For example, at a given scale, the algorithms with a set of selected parameters may successfully detect simple targets such as airplanes and cars; and may fail to detect complex targets, such as airports and parking lots. Therefore, one may need to employ more than one segmentation output obtained at multiple scales to extract various types of targets. Additionally, depending on the target types, one may need to employ more than one set of features in the segmentation algorithms.

In multi-spectral image segmentation problems, it is difficult to find an optimal segmentation algorithm that covers all the spectral bands. Some objects are recognized on specific spectral bands, whereas other objects may require the processing of different bands together. For example, the algorithms with a set of selected parameters may successfully detect objects such as water and shadow in the near-infrared (NIR) band, but may fail to detect objects which provide color or textural information, such as farms and buildings. Therefore, one may need to employ more than one segmentation output obtained from multiple spectral bands to extract various types of objects.

In this chapter, the aforementioned problems have been studied using decision fusion approaches with unsupervised and semi-supervised learning. For this purpose, first a consensus segmentation problem is formalized as an unsupervised segmentation fusion problem. Then, a new segmentation fusion method, called Segmentation Fusion (SF), is introduced based on a consensus clustering algorithm, called Stochastic Filtered Best One Element Move (Filtered Stochastic BOEM) [107]. In the base-layer of the fusion method, different segmentation algorithms or a single segmentation algorithm with a set of different parameters are employed and a set of segmentation outputs (i.e. decisions of segmentation algorithms) is obtained. Then, a fusion strategy is designed by adapting the Filtered Stochastic BOEM method to fuse the decisions of base-layer segmentation algorithms.

In the suggested Segmentation Fusion algorithm, some of the segments in the segmentation decision set are expected to represent target objects in order to be able to obtain a consensus segmentation which represents the target objects. This assumption of the proposed SF algorithm is analyzed with various examples on real-world images. In the examples, complex objects, such as airports, are segmented using two well-known remote sensing image segmentation algorithms, Mean Shift [144] and Recursive Hierarchical Segmentation (RHSEG) [166, 167, 168]. Various sets of segmentations are obtained by implementing Mean Shift and RHSEG with different parameters on the aerial images in order to observe the situations in which a consensus segmentation is appropriate to obtain a segmentation that represents target objects.

The proposed Segmentation Fusion algorithm is an unsupervised segmentation algorithm which aims to achieve a consensus on the decisions of the segmentation algorithms. However, there are various problems of the algorithm. First of them is the estimation of algorithm convergence rate parameter β since the convergence analysis of the proposed SF algorithm is a challenge and the algorithm may terminate before converging to an *optimal* solution if β parameter is not selected appropriately. In order to estimate β, we suggest a method to select a β parameter by minimizing the error function of the algorithm in order to achieve a consensus segmentation. A similar strategy is proposed to estimate the number of different segment labels C, i.e. number of segments, in the images which provides a consensus segmentation with minimum error function.

Moreover, one may be interested in a segmentation output which is closer to a target segmentation. For this purpose, some of the segments in the segmentation decision set are expected to represent acquired target objects in the suggested SF. In order to relax this assumption, first the error function of the algorithm should be refined to include these two requirements. Therefore, prior information on the statistical properties of the datasets need to be incorporated using supervision. In addition, side information about a target segmentation output should be used in the unsupervised segmentation fusion algorithm, which leads to a semi-supervised algorithm.

For this purpose, two novel methods are proposed to learn the distance function employed in the SF using prior information about the targets, such as the distributions of pixels or the number of segments in images.

In addition, *contributions* of the base-layer segmentation algorithms to decision fusion are measured as the error values induced by their decisions to the error function of the segmentation fusion problem. Then, the weighted distance learning problem is defined as a weighted decision (i.e. segmentation) fusion problem. To solve this problem using supervision, performance validation indices are employed to compute the weights [169].

Finally, an algorithm called Semi-supervised Segmentation Fusion (SSSF) is introduced for fusing the segmentation outputs (decisions) of base-layer segmentation algorithms by incorporating the prior information about the data statistics and side-information about the content into the Segmentation Fusion algorithm. In the SSSF, this is accomplished by extracting the available side information about the targets, such as defining the memberships of pixels for the segments which represent a specific target in images. For this purpose, the SF algorithm is reformulated with a set of constraints to incorporate the side information about the pixel-wise relationships.

Based on the distance learning methods for SF, a new distance function is defined for the Semi-supervised Segmentation Fusion by assigning weights to each segmentation. In order to compute the *optimal* weights, the median partition (segmentation) problem is converted into a convex optimization problem. The side information which represents the pixel-wise segmentation membership relations defined by must-link and cannot-link constraints are incorporated in an optimization problem and in the structure of distance functions. In addition, sparsity of the weights are used in the optimization problem for decision *selection*.

Various weighted cluster aggregation methods have been used in the literature [103, 170, 109, 104]. Unlike these methods, the proposed approach and the algorithms enable learning both the structure of the distance function, the pixel-wise relationships and the *contributions* of the decisions of the segmentation algorithms from the data by solving a single optimization problem using semi-supervision.

In the next section, the proposed segmentation fusion method, which generates a consensus among the segmentation outputs is introduced with a brief overview of Filtered Stochastic BOEM problem. The suggested method is examined in various examples in Section 5.2. The assumptions of segmentation fusion algorithms are analyzed to explain the motivation for distance learning and incorporating supervision to the SF in Section 5.3. Algorithms for distance learning, and parameter and optimal segment number estimation are given in Section 5.6, 5.7, 5.5 and 5.4. Weighted decision fusion for segmentation algorithm is given in Section 5.8. Semi-supervised Segmentation Fusion algorithm is introduced in Section 5.9. Computational complexities of the algorithms are analyzed in Section 5.10. Section 5.11

concludes the chapter. Experimental analyses of the algorithms are given in the next chapter.

5.1 Segmentation Fusion using Filtered Stochastic Optimization

Filtered Stochastic BOEM [107] is a consensus clustering algorithm which approximates a solution to the Median Partition Problem [18] by integrating Best One Element Move (BOEM) [18] and Stochastic Gradient Descent (SGD) [171]. In this work, Segmentation Fusion is employed using the filtered stochastic optimization method to solve the segmentation fusion problem which is defined below.

In the proposed segmentation fusion method, an image $\mathbb{I}$ is fed to J different base-layer segmentation algorithms SA_j, $j = 1, 2, \ldots, J$. Each segmentation algorithm is employed on $\mathbb{I}$ to obtain a set of segmentation outputs $S_j = \{s_i\}_{i=1}^{n_j}$ where $s_i \in A^N$ is a segmentation (partition) output, A is the set of segment labels with N pixels, $|A| = C$ different segment labels, and a distance function $d(\cdot, \cdot)$. Note that A^N is the class of all segmentations of finite sets with C different segment labels in the image $\mathbb{I}$.

For instance, an image $\mathbb{I}$ of size 100×100 has $N = 10^4$ pixels. If a Mean Shift algorithm (SA_1) is implemented with $h_s = 1, h_r = 1, minArea = 100$ and $h_s = 2, h_r = 3, minArea = 100$ on $\mathbb{I}$, then we have $n_1 = 2$ and $S_1 = \{s_1, s_2\}$ where s_1 is the segmentation obtained from Mean Shift with $(h_s = 1, h_r = 1, minArea = 100)$ and s_2 is the segmentation obtained from Mean Shift with $(h_s = 2, h_r = 3, minArea = 100)$. If $C = 5$ different segment labels, e.g. background, car, parking area, building and road, are used to label the segments, and the pixels in the segments, then the segmentations s_1 and s_2 are realizations observed in 5^{10^4} tuples. Moreover, another segmentation algorithm SA_2, such as k-means, can be implemented with $k = 5$ providing a segmentation s_3, such that $n_2 = 1$ and $S_2 = \{s_3\}$.

An initial segmentation s is selected from the segmentation set $S = \bigcup_{j=1}^{J} S_j$ consisting of $K = \sum_{j=1}^{J} n_j$ segmentations using algorithms which employ search heuristics, such as Best of K *(BOK)* [18]. Then, a *consensus* segmentation $\hat{s}$ is computed by solving the following optimization problem:

$$\hat{s} = \underset{s}{\operatorname{argmin}} \sum_{i=1}^{K} d(s_i, s) \; .$$

Given two segmentations s_i and s_j, the distance function is defined as the *Symmetric Distance Function (SDD)* given by $d(s_i, s_j) = N_{01} + N_{10}$, where N_{01} is the number of pairs co-segmented in s_i but not in s_j, and N_{10} is the number of pairs co-segmented in s_j but not in s_i [18]. In order to normalize SDD such that segmentation outputs are compared with different segmentations K, a normalized form of SDD which is called *Average Sum of Distances (Average SOD)* [18]

$$SOD = \frac{2 \sum_{i=1}^{K} d(s_i, s)}{KN(N-1)} \quad , \tag{5.1}$$

is used.

At each iteration of the optimization algorithm, a new segmentation is computed. Specifically, a segmentation s is randomly selected from the segmentation set. Then, the best one element

move of the current segmentation s is computed with respect to the objective of the optimization and applied to the current segmentation to generate a new segmentation. If there is no improvement on the best move, the current segmentation is returned by the algorithm.

The Best One Element Move (BOEM) of segmentation s is defined as

$$\Delta s = \frac{\partial \sum_{i=1}^{K} d(s_i, s)}{\partial s},$$

and can be evaluated by

$$\Delta s_t = \frac{\partial H_t}{\partial s_t},$$

where

$$H_t = \sum_{i=1}^{K} d(s_i, s_t)$$

is the objective at time t. Using the assumption that single element updates do not change the objective function, H_t can be approximated by H_{t-1} with a scale parameter $\beta \in [0, 1]$. Then,

$$\Delta s_t = \frac{\partial}{\partial s_t}(\beta H_{t-1} + d(s_{i'}, s_t)) \ ,$$

where $s_{i'}$ is the randomly selected segmentation for updating the current BOEM. If an $N \times C$ matrix $[H]$ is defined such that the n^{th} row and the c^{th} column of the matrix, $[H]_{nc}$, is the updated value of H obtained by switching n^{th} element of s to the c^{th} segment label, the move can be approximated by

$$\underset{n,c}{\operatorname{argmin}} \ \beta[H_{t-1}]_{n,c} + [d(s_{i'}, s_t)]_{n,c} \ , \tag{5.2}$$

if $s_{i'}$ is selected for updating s_t at time t, $\forall i = 1, 2, \ldots, N$, $\forall c = 1, 2, \ldots, C$.

input : Input image $\mathbb{I}$, $\{SA_j\}_{j=1}^{J}$, T.
output: Output segmentation O
1 Run SA_j on I to obtain $S_j = \{s_i\}_{i=1}^{n_j}$, $\forall j = 1, 2, \ldots, J$.;
2 At $t = 1$, initialize s and $[H_t]$;
for $t \leftarrow 2$ **to** T **do**
3 Randomly select one of the segmentation results with the index $i' \in \{1, 2, \ldots, K\}$;
4 $[H_t] \leftarrow \beta[H_t] + [d(s_{i'}, s)]$;
5 Find Δs by solving $\underset{n,c}{\operatorname{argmin}} \beta[H_t]_{n,c}$;
6 $s \leftarrow s + \Delta s$;
7 $t \leftarrow t + 1$;
end
8 $O \leftarrow s$;

Algorithm 6: Segmentation Fusion.

In the proposed Segmentation Fusion (SF) Algorithm, $[H_t]$ is initialized at $t = 1$. Until t reaches a given termination time T, the segmentation s is updated. A segmentation is randomly selected from a pseudo-random permutation of the numbers $1, 2, \ldots, K$ until all the segmentations in $s_1, s_2, \ldots, s_K$ are traversed. Then, a new segmentation is generated and this operation is repeated until all of the permutations are traversed. $[H_t]$ is updated by aggregating $[d(s_{i'}, s)]$ with the scaled $\beta[H_t]$.

$\beta \in [0,1]$ controls the convergence rate and the performance of the algorithm. If $\beta = 0$, the algorithm employs pure stochastic BOEM and the algorithm is memoryless. If $\beta = 1$, the algorithm *forgets* slowly. However, Zheng, Kulkarni and Poor [107] reported that the algorithm may perform worse if β is on either end of $[0,1]$. Selection of the *optimal* β values for segmentation fusion is a very crucial problem. An optimization method is suggested to find an *appropriate* β as explained in Section 5.5. After $[H_t]$ is updated, Δs is computed in order to update s. The algorithm is iterated until the termination criterion is achieved.

5.2 Examples to Analyze the Segmentation Fusion Algorithm

The aim of this subsection is to investigate various properties of the suggested Segmentation Fusion (SF) algorithm, empirically. For this purpose, some real-world examples are first segmented by popular segmentation algorithms, namely Mean Shift and RHSEG. Then the outputs of these algorithms are fused to generate a consensus over these algorithms.

In the examples, image datasets which are collected using Google Maps at different resolutions from 8m to 16m are processed. The proposed algorithm is examined by fusing the decisions of Mean Shift, RHSEG and Mean Shift + RHSEG algorithms. Complex targets such as substations, airports and harbors in the images are considered.

An EDISON C++ implementation of Mean Shift is used in the examples. Mean Shift extracts a 5-dimensional feature vector from the given image which consist of R, G, B channels, and X and Y spatial coordinates. Kernel density estimation is employed with different parameters $(h_s, h_r, minArea)$ in heterogeneous feature spaces. An overview of the algorithm is given in Section 4.1.1.1.

The hierarchical segmentation algorithm (HSEG) [166], is a popular segmentation method that merges segments at different levels of detail. Tilton provides an algorithmic description of the HSEG as follows [56];

1. **Initialization:** Assign a segment label to each pixel in the image.
   - If a pre-segmentation output is available, use the segment labels obtained from the pre-segmentation output.
   - Otherwise, assign a different label to each different pixel.
2. Compute the distance between all spatially adjacent segments using a distance function which defines the (dis)similarity between the segments. Then, merge the segments with the smallest distance.
3. If there are no more segments that will be merged, then terminate the algorithm. Otherwise, return to the previous step.

RHSEG is a recursive approximation of HSEG that recursively subdivides the image data for more efficient analysis [167]. In [168], RHSEG is employed with the other segmentation algorithms in an ensemble of classifiers for the classification of multi-spectral data. In RHSEG, individual pixels or segments are labelled initially with different labels. Then, a (dis)similarity

value, such as minimizing entropy change, spectral information divergence or spectral angle mapper, is computed between all pairs of spatially adjacent segments or pixels [172].

In RHSEG, a spectral weight parameter $0 \leq sp_w \leq 1$ is introduced by the user in order to merge the spatial or spectral neighbors. If $sp_w = 0$, only the spatially neighboring segments or pixels are merged. For $0 < sp_w \leq 1$, the merging of spatially adjacent or non-adjacent pixels and the segments are balanced with $\frac{1}{sp_w}$. This process is recursively employed until convergence is achieved. An algorithmic description of RHSEG is given below [56];

1. Given an input image $\mathbb{I}$,
   - Compute the number of levels of recursion $level_r$ such that each spatial dimension of $\mathbb{I}$ can be evenly divided by $2^{level_r - 1}$.
   - Set the level index $level = 1$.
2. Call rhseg($\mathbb{I}$, $level$) which is defined in the following steps;
   (a) If $level = level_r$, go to step (c). Otherwise, divide $\mathbb{I}$ into 4 sub-images $\{\mathbb{I}_i\}_{i=1}^4$ and call rhseg($\mathbb{I}_i$, $level + 1$) for each sub-image $\mathbb{I}_i$.
   (b) Call the previous step 4 times and collect the segmentation results.
   (c) If $level < level_r$, initialize the segmentation with the segmentation results obtained from the previous step. If $level = level_r$, exit. Otherwise, do the following and exit;
   - For each segment, find a set $candidate\_segment$ of segments that are more similar to each other than a value defined by $F_{region} \times sp_w \times merge_{max}$, where F_{region} is a user defined value and $merge_{max}$ is the maximum of the distances computed in the previous steps and used for merging for $sp_w > 0$.
   - For each segment with $|candidate\_segment| > 0$, select the pixels in the segment that are more similar to segments in $candidate\_segment$ than to their current segment by a factor F_{split}. If $sp_w = 0$, assign these pixels to the most similar segment. Otherwise, first split these pixels from their current regions. Then, merge them by running HSEG using the segments of the pixels that are split out from and the segments in $candidate\_segment$.
3. Run HSEG on $\mathbb{I}$ using the output of the previous step as a pre-segmentation output.

In the examples, an implementation of RHSEG provided by Tilton is used [172]. $sp_w = 0$ is employed with region sum features, while the other parameters are used as default values, e.g. $F_{region} = F_{split} = 1.5$ [56]. Since our goal for the analyses on the examples is to investigate the variation of the performance of SF on various different image segmentations, the parameters of the Mean Shift algorithm are randomly selected. The SF algorithm is implemented with $\beta = 0.6, 0.7, \ldots, 0.9$ and $\beta = 0.99$ with the termination time T equal to the partition size [107]. The segmentations which provide the minimum Average SOD values are given.

5.2.1 Performance Measures used in the Examples

Since the proposed Segmentation Fusion algorithm is based on unsupervised learning, it is assumed that ground truth of the objects in the images are not available in the examples.

Therefore, two types of performance measures for unsupervised segmentation are used. First, the distance between a base-layer segmentations s_i and segmentation fusion output O is measured using Average Sum of Distances (SOD), which is defined in (5.1) as

$$SOD = \frac{2\sum_{i=1}^{K} d(s_i, O)}{KN(N-1)} .$$

In addition, the (dis)similarity between a segmentation algorithm output s_i and segmentation fusion output O is measured using i) Rand Index (RI) and ii) Adjusted Rand Index (ARI), which provide information about the probability of agreement, iii) Mirkin's Index (MI) which provides information about the probability of disagreement, and iv) Hubert's Index (HI) which provides information about the difference between probability of agreement and probability of disagreement [19].

Mathematically speaking, given two segmentations s_i and s_j, Rand Index RI is defined as

$$RI(s_i, s_j) = 1 - \frac{d(s_i, s_j)}{\binom{N}{2}},$$

where

$$d(s_i, s_j) = N_{10} + N_{01} = \binom{N}{2} - (N_{00} + N_{11})$$

and N_{01} is the number of pairs co-segmented in s_i but not in s_j, N_{10} is the number of pairs co-segmented in s_j but not in s_i, N_{11} is the number of pairs co-segmented in both s_i and s_j, i.e. the number of pairs that are in the same segment, and N_{00} is the number of pairs that are in different segments.

On the other hand, Adjusted Rand Index (ARI) [173] is defined as

$$ARI(s_i, s_j) = \frac{\sum_{n_i=1}^{K_i}\sum_{n_j=1}^{K_j}\binom{\mathfrak{N}_{ij}}{2} - a_{ij}}{\frac{1}{2}(a_i + a_j) - a_{ij}} , \tag{5.3}$$

where $\mathfrak{N}(i)$ is defined as the number of pixels in the i^{th} segment of s_i, and $\mathfrak{N}_{ij}$ is defined as the number of pixels in both the i^{th} segment of s_i and the j^{th} segment of s_j, $a_i = \sum_{n_i=1}^{K_i}\binom{\mathfrak{N}(i)}{2}$, $a_j = \sum_{u_j=1}^{K_j}\binom{\mathfrak{N}(j)}{2}$ and $a_{ij} = \frac{2a_i a_j}{N(N-1)}$.

When the output segmentation O and a base-layer segmentation are identical, the ARI and the RI are equal to 1. Moreover, the ARI equals 0 when the RI equals its expected value.

Finally, for the sake of completeness, we provide the definitions of two more popular indexes, Mirkin's (MI), which is defined as

$$MI(s_i, s_j) = 1 - \frac{N_{00} + N_{11}}{\binom{N}{2}},$$

and Hubert's Index which is defined as the difference between Rand Index and Mirkin's Index, $HI = RI - MI$. A more detailed mathematical treatment of the indices is given in Section 5.6. Since Mirkin's and Hubert's indexes are consistent with RI, we skip interpreting the values obtained for these indexes in the following analyses.

A summary of the examples is presented in Table 5.1. In the examples, the results of Mean Shift and RHSEG segmentation are fed into the SF. The results of Mean Shift are fused in

Table 5.1: Summary of The Examples.

Figure Name	**Image Size**	**Seg. Algorithm**	**Average** *SOD*
Figure 5.1	1060 x 922	Mean Shift	3.2
Figure 5.2	1076 x 922	Mean Shift	8.7
Figure 5.3	1024 x 768	Mean Shift	1.4
Figure 5.4	1030 x 850	Mean Shift	5.8
Figure 5.5	1455 x 794	RHSEG	10.3
Figure 5.6	1455 x 794	Mean Shift and RHSEG	8.9

Figure 5.1-5.4, of RHSEG are fused in Figure 5.5 and of both of the algorithms are fused in Figure 5.6.

The *SOD* values, which are computed in the examples, are provided in Table 5.1. Average *SOD* values can be considered as the similarity between the output of the SF algorithm, O, and the segmentation outputs, s_i, $i = 1, 2, \ldots, K$. For instance, the distances between output images of SF and the base-layer segmentation algorithms, which are shown in Figure 5.3, are more smaller than the distances computed in the other experiments. Therefore, it can be stated that SF algorithm provides a segmentation output which is *similar* or *closer* to the base-layer segmentation algorithm outputs in the example given in Figure 5.3 than the other examples given in the other images.

5.2.2 Examples using Mean Shift Segmentation

In this section, the proposed SF method is employed on the images consisting of substations and airports to fuse the outputs of the Mean Shift segmentation algorithm obtained with different parameters.

5.2.2.1 Examples on Substation Images

In this subsection, Mean Shift segmentation algorithm is applied on several remotely sensed images with $h_s = 3$ and $h_r = 3$. Different *minArea* values are used to obtain different segmentation outputs. In the first example, the segmentation outputs of base-layer segmentation algorithms and the SF algorithm are displayed in Figure 5.1. The performances indices of the algorithms are given in Table 5.2.

The original image is shown in Figure 5.1.a. The segmentation outputs of the Mean Shift algorithm are shown in Figure 5.1.b to Figure 5.1.i and the output of the SF algorithm is shown in Figure 5.1.j. Note that, four segmentation outputs in Figure 5.1.b to Figure 5.1.e provide over-segmented regions for the substation and an under-segmented region is shown in Figure 5.1.i.

Among many Mean Shift segmentation outputs run with different parameters, the most similar one to the output of the suggested SF algorithm is shown in Figure 5.1.f where both *RI* and *ARI* indexes are maximum. Visual inspection among various outputs reveals that Figure 5.1.f extracts the whole substation region. The next similar segmentation outputs are shown in Figure 5.1.g and Figure 5.1.h. The over-segmented and the under-segmented outputs, which

are shown in Figure 5.1.b and Figure 5.1.i, respectively, are the most dissimilar segmentation outputs.

Table 5.2: Performances for the segmentation outputs shown in Figure 5.1 with Average SOD = 3.2.

Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**	**(i)**
ARI	0.57	0.61	0.61	0.64	0.71	0.66	0.64	0.52
RI	0.89	0.91	0.90	0.92	0.98	0.96	0.96	0.78
MI	0.11	0.09	0.10	0.08	0.02	0.04	0.04	0.22
HI	0.78	0.82	0.80	0.84	0.96	0.92	0.92	0.56

The segmentation outputs of the algorithms, which are employed on another image, are shown in Figure 5.2, and the performances are given in Table 5.3. The original image is shown in Figure 5.2.a. The segmentation outputs of the Mean Shift algorithm are shown in Figure 5.2.b to Figure 5.2.i and the output of the SF algorithm is shown in Figure 5.2.j.

When RI values, which are given in Table 5.3, are analyzed, it is observed that the output of the SF algorithm is similar to the outputs of over-segmented images, which are Figure 5.2.b, Figure 5.2.c and Figure 5.2.d. If ARI values are analyzed, base-layer segmentation outputs, which are shown in Figure 5.2.d and Figure 5.2.e, are more similar to the SF output, than the other segmentation outputs. For this particular example, RI does not provide a visually meaningful performance metric due ti the violation of equal segment number assumption in both Mean Shift and Segmentation Fusion. On the other hand, ARI, which is robust to the violation of this assumption, enables us to select the visually meaningful Mean Shift output (see Figure 5.2.d and Figure 5.2.e). On the other hand, the most dissimilar segmentation outputs, which are Figure 5.2.g, Figure 5.2.h and Figure 5.2.i, are the most under-segmented results with highest $minArea$ values, such as $minArea$ = 50000, $minArea$ = 100000 and $minArea$ = 200000, respectively.

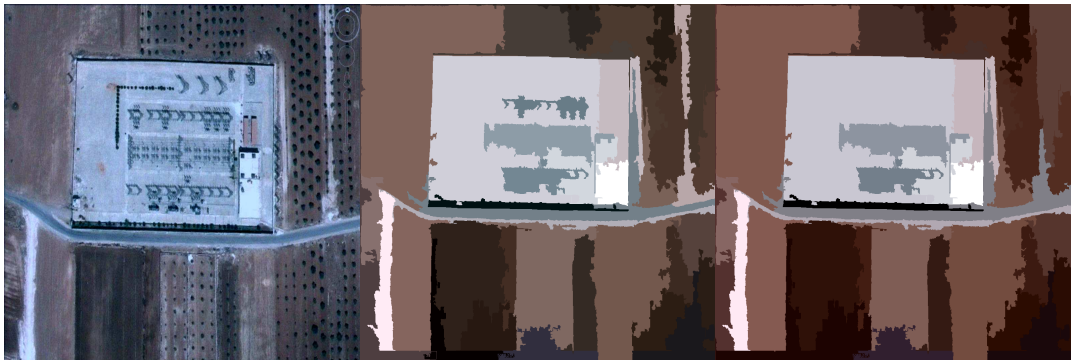
Table 5.3: Performances for the segmentation outputs shown in Figure 5.2 with Average SOD = 8.7.

Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**	**(i)**
ARI	0.40	0.39	0.42	0.42	0.41	-0.04	0.01	0.00
RI	0.74	0.74	0.74	0.73	0.70	0.57	0.48	0.27
MI	0.26	0.26	0.26	0.27	0.30	0.43	0.52	0.73
HI	0.48	0.47	0.48	0.46	0.40	0.13	-0.03	-0.47

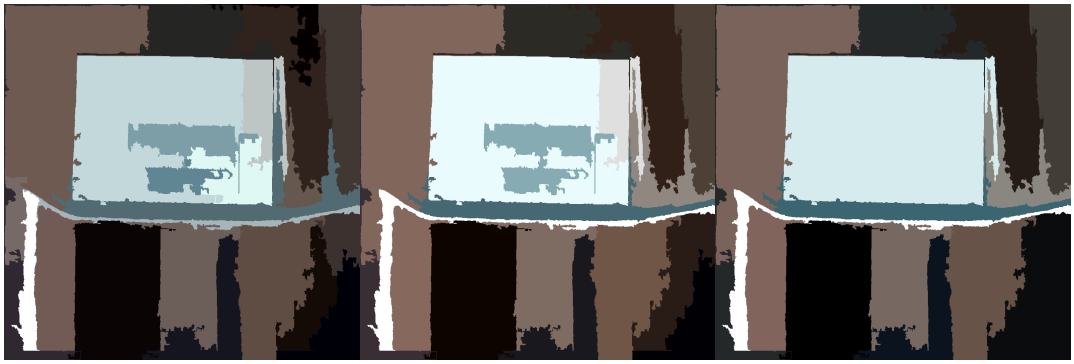
5.2.2.2 Examples on Airport Images

In this section, Mean Shift algorithm with h_s = 1 and h_r = 1 is applied on an airport image shown in Figure 5.3.a. The performance values are given Table 5.4. The segmentation outputs of the Mean Shift algorithm are shown in Figure 5.3.b to Figure 5.3.k and the output of the SF algorithm is shown in Figure 5.3.l. Note that the segmentation outputs in Figure 5.3.b to Figure 5.3.g provide over-segmented regions for the airport and two under-segmented regions are shown in Figure 5.3.j and Figure 5.3.k.

RI values, which are given in Table 5.4, show that the over-segmented images, which are given

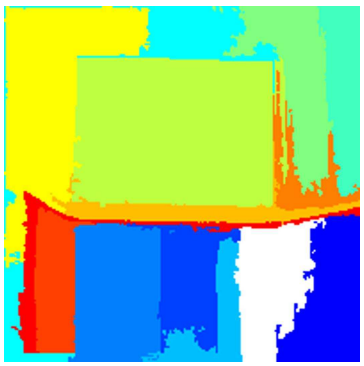


(a) Original Image (b) $minArea$=7000 (c) $minArea$=8000



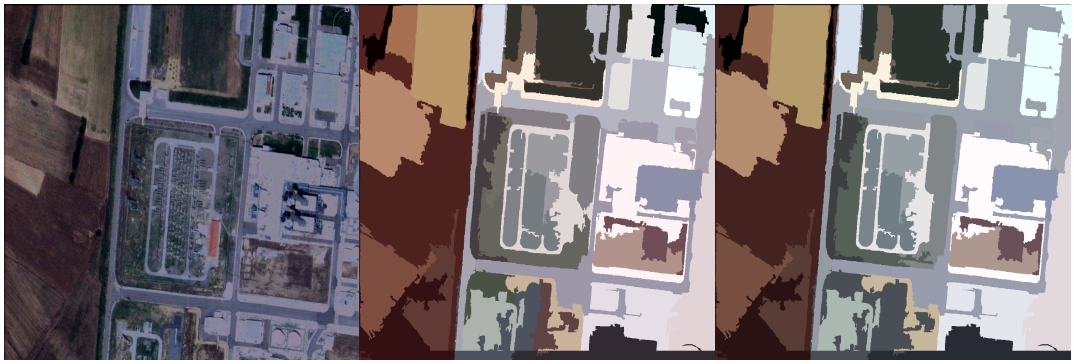
(d) $minArea$=10000 (e) $minArea$=15000 (f) $minArea$=20000

(g) $minArea$=30000 (h) $minArea$=40000 (i) $minArea$=200000

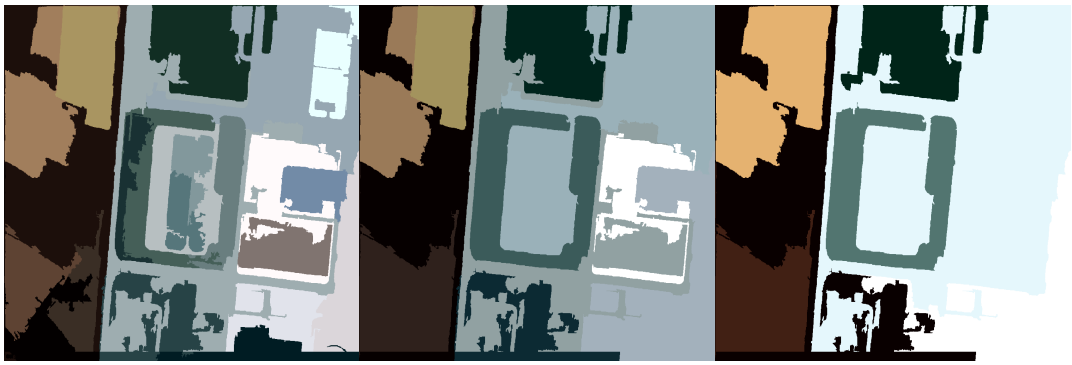


(j) Output of SF

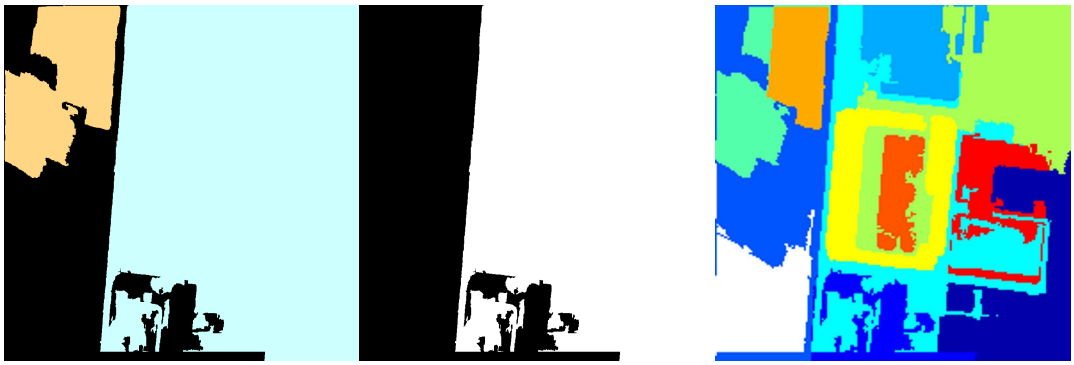
Figure 5.1: Examples with Mean Shift segmentation algorithm, h_s = 3 and h_r = 3 with different $minArea$ values.



(a) Original Image (b) *minArea*=8000 (c) *minArea*=10000



(d) *minArea*=15000 (e) *minArea*=40000 (f) *minArea*=50000



(g) *minArea*=100000 (h) *minArea*=200000 (i) Output of SF

Figure 5.2: Examples with Mean Shift segmentation algorithm, h_s = 3 and h_r = 3 with different *minArea* values.

in Figure 5.3.b to Figure 5.3.f, are the more similar to the output of the SF than the other segmentation outputs. On the other hand, ARI values computed on these images (Figure 5.3.b to Figure 5.3.f) are smaller than the other images. As mentioned before, this behaviour is observed because of two reasons. First the number of segments and the number of pixels in the segments, which are observed in these images, are much more different than the segment numbers and sizes observed in other images. Therefore, the assumption of RI measures, which is the uniform distribution of the samples in the segments, is not valid in these images. Second, the differences between the expected values and the maximum values of the distances among the segmentation outputs of base-layer and SF decrease as $minArea$ values increase in these images. For this reason, ARI provides a better metric to measure the similarity between the fused segmentation output of SF and Mean Shift output.

Table 5.4: Performances for the segmentation outputs shown in Figure 5.3 with Average SOD = 1.4.

Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**	**(i)**	**(j)**	**(k)**
ARI	0.41	0.41	0.41	0.47	0.51	0.53	0.53	0.54	0.52	0.51
RI	0.82	0.82	0.82	0.82	0.82	0.81	0.81	0.81	0.78	0.65
MI	0.18	0.18	0.18	0.18	0.18	0.19	0.19	0.19	0.22	0.35
HI	0.64	0.64	0.64	0.64	0.63	0.63	0.61	0.62	0.57	0.31

The segmentation outputs and the performance values of the algorithms, which are employed on another airport image are given in Figure 5.4 and Table 5.5, respectively. The original image is shown in Figure 5.4.a. The segmentation outputs of the Mean Shift algorithm are shown in Figure 5.4.b to Figure 5.4.k and the output of the SF algorithm is shown in Figure 5.4.l. Note that six segmentation outputs in Figure 5.4.b to Figure 5.4.g provide over-segmented regions for the substation and two under-segmented regions are shown in Figure 5.4.j and Figure 5.4.k.

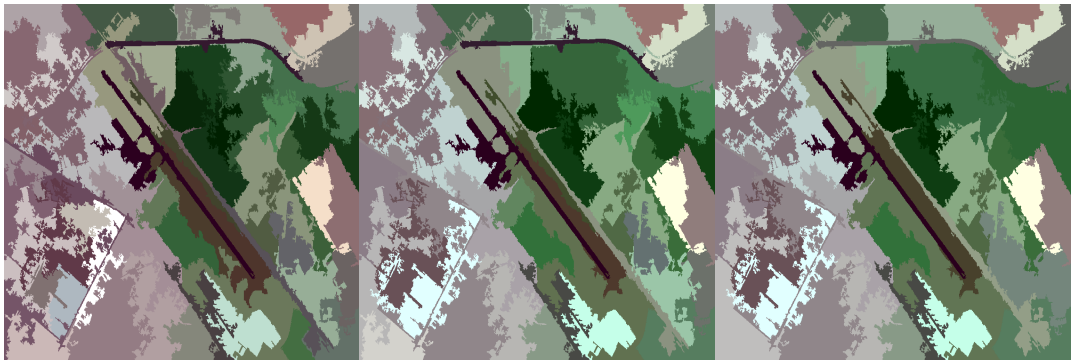
In Table 5.5, the SF output is closer to the segmentation output with $minArea$ = 8000, which is shown in Figure 5.4.i. Unlike the results in Table 5.4, ARI values increase as RI values increase in Table 5.5. This is due to the fact that the statistical properties of the segmentations, such as the number of segments and the pixels in the segments, do not change drastically, as $minArea$ increases.

Table 5.5: Performances for the segmentation outputs shown in Figure 5.4 with Average SOD = 5.8.

Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**	**(i)**	**(j)**
ARI	0.50	0.51	0.60	0.61	0.64	0.69	0.74	0.82	0.40
RI	0.80	0.80	0.80	0.81	0.86	0.86	0.88	0.94	0.65
MI	0.20	0.20	0.20	0.19	0.12	0.12	0.10	0.00	0.37
HI	0.60	0.60	0.60	0.62	0.74	0.74	0.78	0.94	0.28

(a) Original Image (b) *minArea*=500 (c) *minArea*=1000

(d) *minArea*=1500 (e) *minArea*=2000 (f) *minArea*=3500

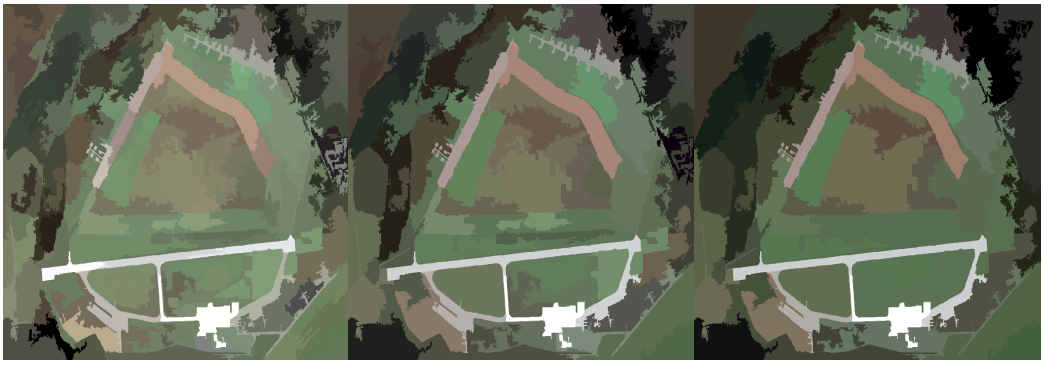


(g) *minArea*=5000 (h) *minArea*=6500 (i) *minArea*=8000

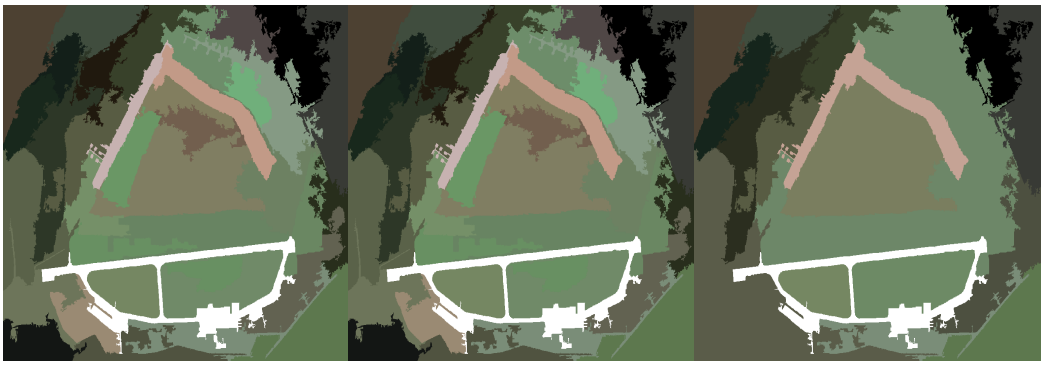
(j) *minArea*=20000 (k) *minArea*=40000 (l) Output of SF

Figure 5.3: Examples with Mean Shift segmentation algorithm, h_s = 1 and h_r = 1 with different *minArea* values.

(a) Original Image (b) *minArea*=500 (c) *minArea*=1000

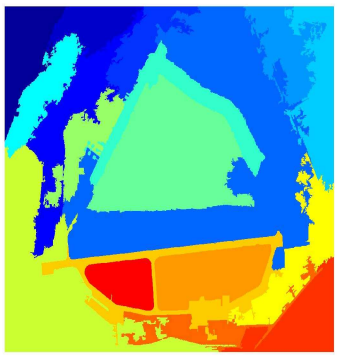


(d) *minArea*=1500 (e) *minArea*=2000 (f) *minArea*=3500



(g) *minArea*=5000 (h) *minArea*=6500 (i) *minArea*=8000

(j) *minArea*=20000



(k) Output of SF

Figure 5.4: Examples with Mean Shift segmentation algorithm, h_s = 1 and h_r = 1 with different *minArea* values.

5.2.3 Examples using RHSEG Segmentation

In this section, RHSEG algorithm is implemented with the parameters introduced in the former sections and several segmentation levels, which are fed into the SF algorithm, are obtained. In this example, a harbor image, which is shown in Figure 5.5.a is used. The segmentation outputs of RHSEG are shown in Figure 5.5.b to Figure 5.5.h and the output of the SF is shown in Figure 5.5.i. The performance values obtained in the example are given in Table 5.6.

In Figure 5.5, the number of segments decreases as the segmentation level increases. However, there is no linear change in ARI or RI values as the level increases, which are given in Table 5.6. On the other hand, it is observed the segmentation output of the proposed algorithm is closer to the segmentation output obtained at $level = 6$ (Figure 5.5.d), when the performance values given in the table are considered.

Table 5.6: Performances for the segmentation outputs shown in Figure 5.5 with Average $SOD = 10.3$.

Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**
ARI	0.51	0.62	0.81	0.62	0.65	0.63	0.57
RI	0.80	0.92	0.95	0.83	0.87	0.90	0.81
MI	0.20	0.08	0.05	0.16	0.13	0.12	0.18
HI	0.60	0.84	0.90	0.67	0.74	0.78	0.63

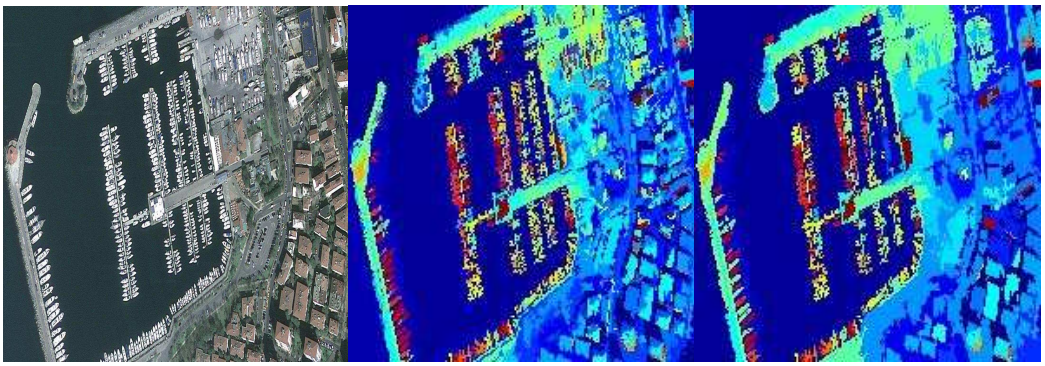
5.2.4 Level Selection and Fusion on Mean Shift and RHSEG Segmentation

In this section, the SF algorithm is employed on the outputs of both Mean Shift and RHSEG segmentation algorithms.

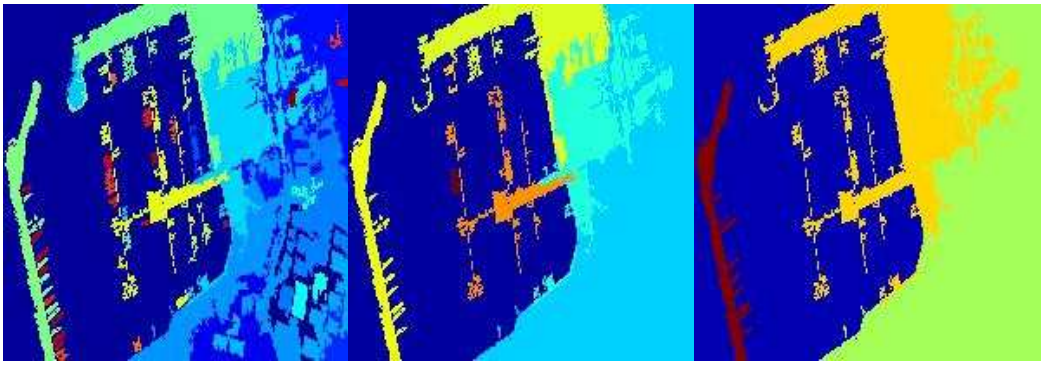
For this purpose, first the harbor image in Figure 5.5.a is segmented with Mean Shift algorithm. The parameters of Mean Shift are selected as $h_s = 3$ and $h_r = 3$ with varying $minArea$ values. The segmentation outputs of the Mean Shift algorithm are shown in Figure 5.6.a- Figure 5.6.h. Then, the SF algorithm is employed on the segmentation outputs of RHSEG algorithm, which are shown in Figure 5.5.b to Figure 5.5.h, together with the outputs of Mean Shift algorithm (Figure 5.6.a - Figure 5.6.h).

The output of the SF algorithm is visualized in Figure 5.6.i. The distance values between the segmentation outputs of SF and RHSEG algorithm are given in Table 5.7, and the outputs of Mean Shift algorithm are given in Table 5.8.

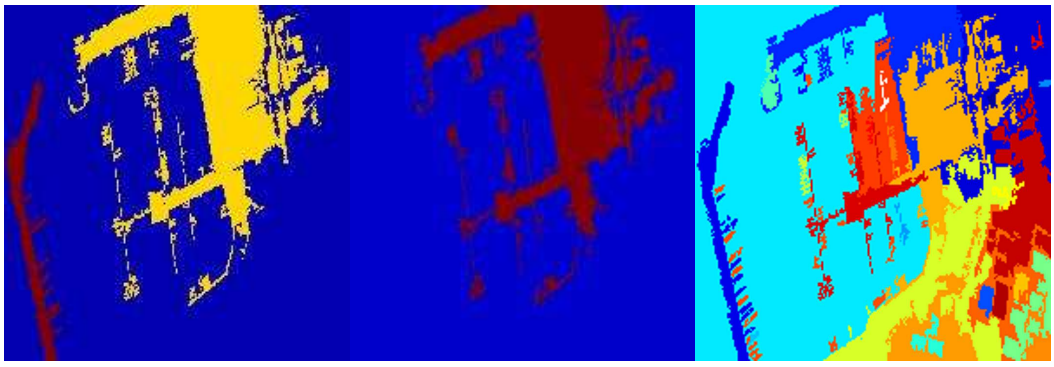
Note that, Average SOD computed for RHSEG segmentations decreases when the RHSEG segmentations are fused with Mean Shift partitions, as given in Table 5.1. In addition, the performance values of some of the segmentation outputs of RHSEG, which are given in Table 5.6, increase in Table 5.7. Since the outputs of Mean Shift algorithm provide the segmented images with the statistical properties similar to the segments in Figure 5.5.b, the output of the SF algorithm is closer to the segmentation output given in Figure 5.5.b, in this example.



(a) Original Image (b) *level* = 2 (c) *level* = 4

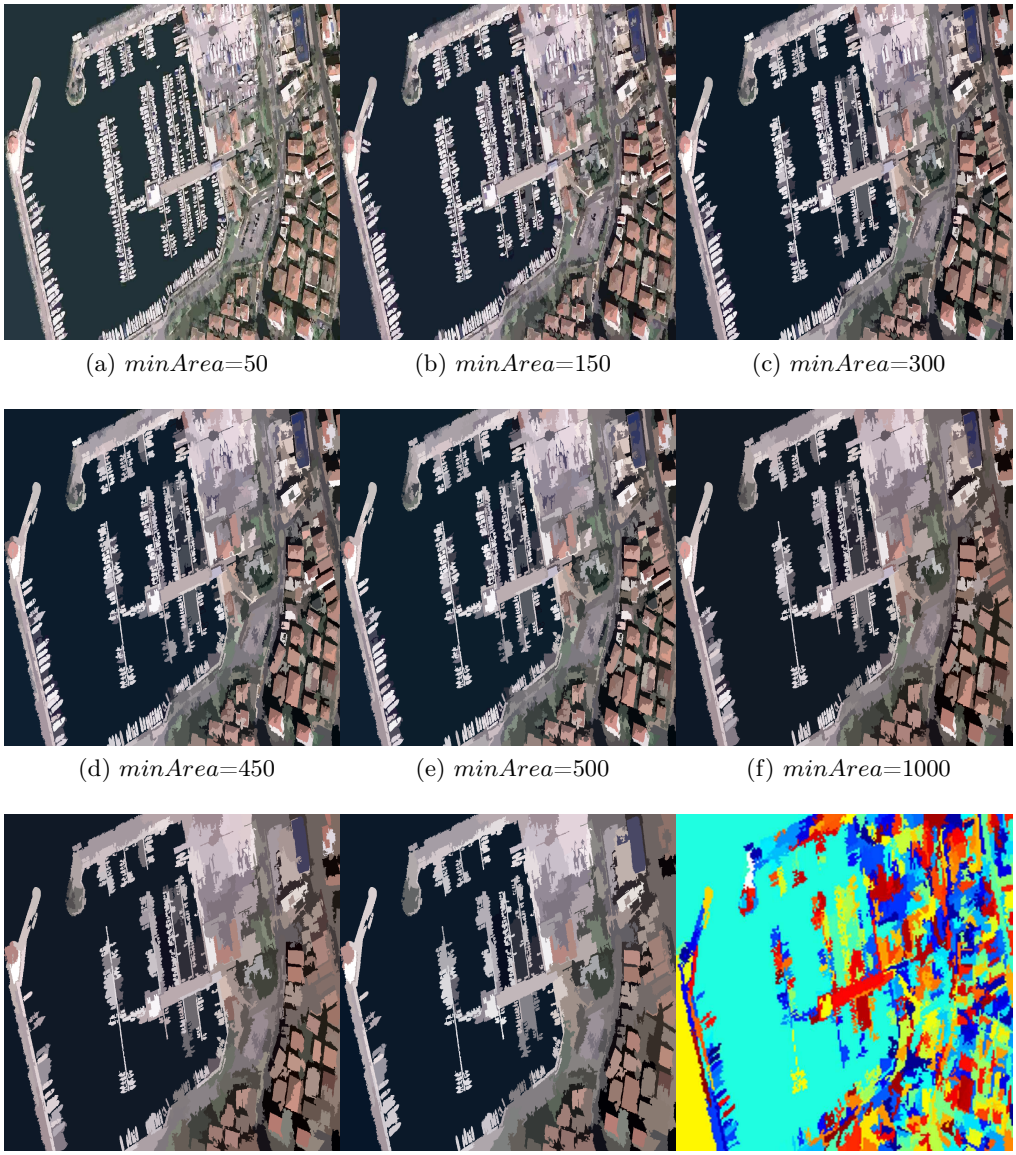


(d) *level* = 6 (e) *level* = 8 (f) *level* = 9



(g) *level* = 10 (h) *level* = 11 (i) Output of SF

Figure 5.5: Examples with RHSEG segmentation algorithm, with different segmentation levels.



(a) $minArea$=50 (b) $minArea$=150 (c) $minArea$=300

(d) $minArea$=450 (e) $minArea$=500 (f) $minArea$=1000

(g) $minArea$=1500 (h) $minArea$=2000 (i) Output of SF with Mean Shift and RHSEG algorithms

Figure 5.6: Examples with Mean Shift and RHSEG segmentation algorithms.

Table 5.7: Performances obtained in the example with Mean Shift and RHSEG segmentation algorithms with Average SOD = 8.9.

RHSEG Segmentations (Figure 5.5)							
Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**
ARI	0.64	0.63	0.60	0.60	0.60	0.56	0.55
RI	0.93	0.92	0.92	0.91	0.91	0.90	0.90
MI	0.07	0.08	0.08	0.09	0.09	0.10	0.10
HI	0.86	0.85	0.83	0.83	0.83	0.81	0.80

Table 5.8: Performances obtained in the example with Mean Shift and RHSEG segmentation algorithms with Average SOD = 8.9.

Mean Shift Partitions (Figure 5.6)							
Performance	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**	**(h)**
ARI	0.54	0.24	0.21	0.20	0.10	0.21	0.08
RI	0.89	0.86	0.84	0.81	0.70	0.85	0.74
MI	0.11	0.14	0.16	0.19	0.30	0.15	0.26
HI	0.78	0.72	0.68	0.61	0.40	0.69	0.49

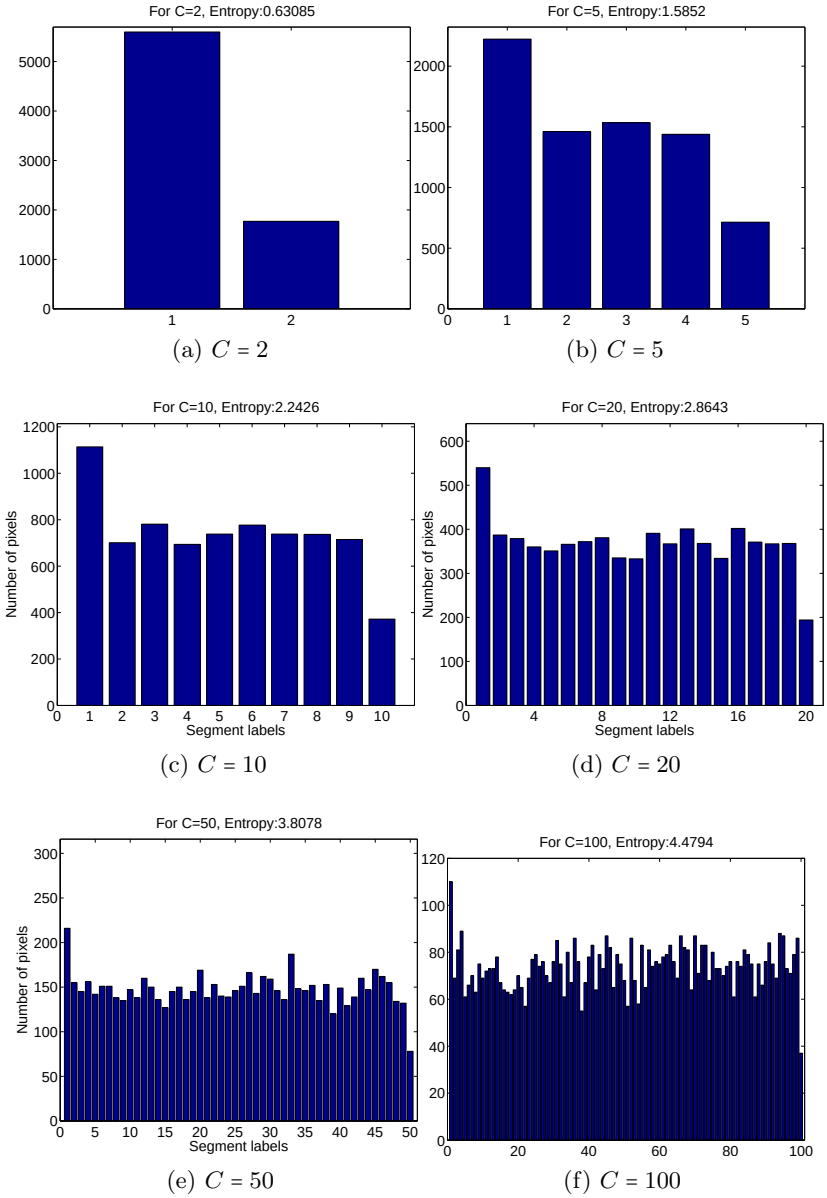
In the above examples, we observe that various assumptions of SF may be invalid in the segmentation of real-world images. For instance, segmentation outputs of different segmentation algorithms may contain different number of segments. Then, under-segmented and over-segmented segmentations are observed at the output of the Segmentation Fusion algorithm. Therefore, the assumption of RI, which is the uniform distribution of the samples in the segments, is not valid. Since RI is used to compute the distance function of SF, violation of the assumption affects the performance of SF. In addition, if the number of segmentations which contain over-segmented or under-segmented images is greater than the number of segmentations which represent the target objects, then SF does not provide a consensus segmentation output which represents the target objects.

In order to address these problems, we, first, analyze the assumption of SF on the distribution of pixels and segments in the segmentations in the following section. Then, we suggest two distance learning methods which employ an adjusted form of RI, i.e. ARI, in the computation of distance functions to overcome the assumptions of RI and SF in Section 5.6 and 5.7. Next, two methods are proposed to estimate the number of different segment labels and algorithm parameter of SF by minimizing the error function of SF in Section 5.5. In order to select the decisions of base-layer segmentation algorithms or fuse them according to their contribution to the performance of SF, a weighted segmentation fusion method is suggested in Section 5.8.

Moreover, prior information on the statistical properties of the data and side information obtained from expert knowledge, such as must-link and cannot-link constraints of segments and pixels, is incorporated into SF by defining the segmentation fusion problem as a convex optimization problem. Then, the convex optimization problem is solved using ADMM [112] method with SF to compute the weights of decisions of base-layer segmentation algorithms for their weighted decision fusion in Section 5.9.

5.3 Analysis of the Assumptions of Segmentation Fusion Algorithm

Segmentation Fusion is an unsupervised segmentation technique where the parameters are randomly selected. However, the consensus generation strategies and rules for parameter selection can be learned from the data, if the prior information or the target labels are available.



(a) $C = 2$ (b) $C = 5$

(c) $C = 10$ (d) $C = 20$

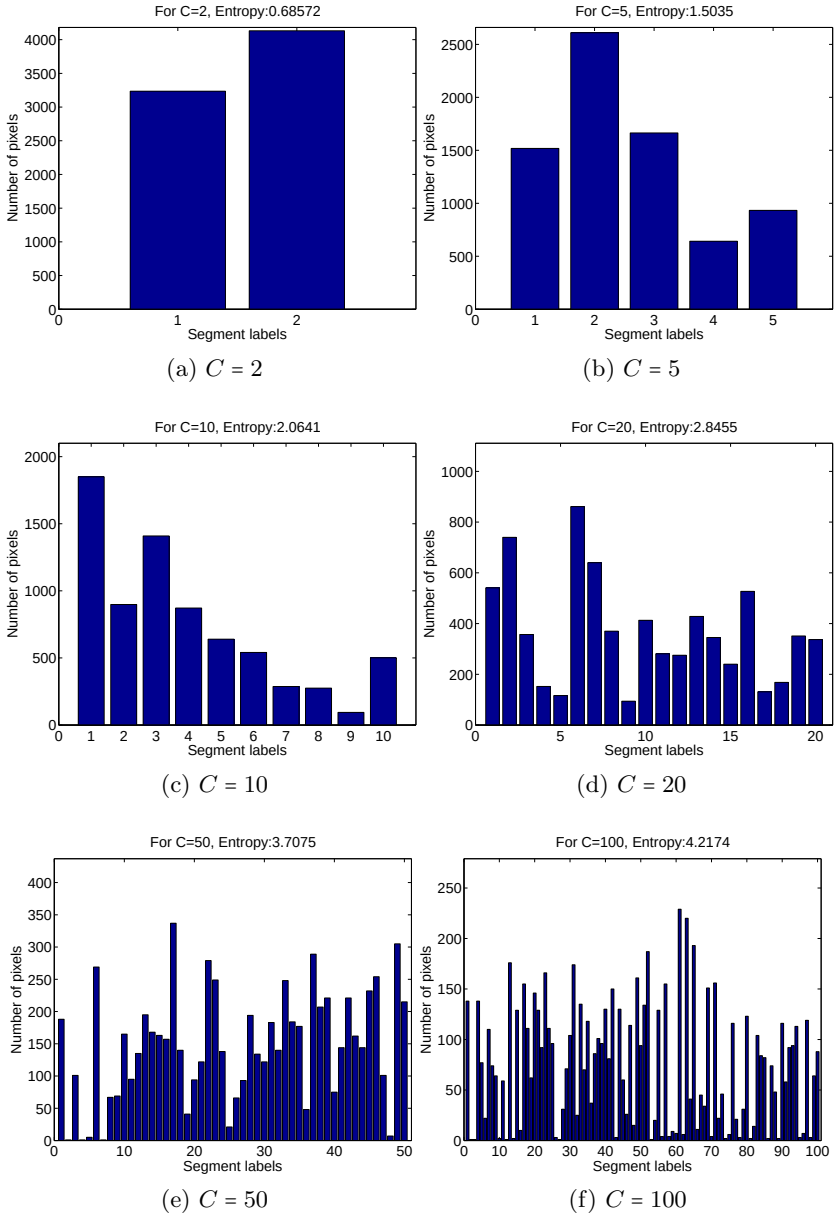
(e) $C = 50$ (f) $C = 100$

Figure 5.7: Distribution of the pixels in segmentation outputs for different C number of segment names, using k-means algorithm.

One of the crucial assumptions of consensus segmentation (clustering) algorithms is that the number of segments (clusters), C are fixed in each segmentation set S_j. However, this assumption may not be satisfied when the segmentation algorithms are implemented for different number of segments. For instance, a segmentation algorithm SA_i can be implemented for the segmentation of buildings and roads in S_i for $C = 2$, and another algorithm SA_j for the

segmentation of building, roads and farms in S_j for $C = 3$.

This assumption leads to inaccurate computations of normalized distances for the consensus among segmentation with different C number of segments. During the computation of Average Sum of Distances (Average SOD) (5.1), it is assumed that the distributions are uniform and $C_i = c, \forall i = 1, 2, \ldots, K$ for a constant c, where K is the total number of segmentations. Additionally, the probability of the disagreement between two different segmentations for co-segmenting two elements is assumed to be a Normal distribution. Therefore, the segmentations are chosen uniformly at random in order to solve (5.1) in SF.



(a) $C = 2$ (b) $C = 5$

(c) $C = 10$ (d) $C = 20$

(e) $C = 50$ (f) $C = 100$

Figure 5.8: Distribution of the pixels in segmentation outputs for different C number of segment names, using Normalized Cuts algorithm.

However, this assumption may fail in some cases. For instance, the k-means algorithm is implemented for different k values in order to produce $C = k$ number of distinct segments for

the segmentation of an aerial image (Figure 5.1) in Figure 5.7 and Normalized Cuts algorithm (see Section 6.1) is implemented for $C = k$ distinct segments in Figure 5.8. The distributions of the pixels for each segment index $c = 1, 2, \ldots, C$, are given in the figures. In Figure 5.7, we observe that the distributions are not uniform for $C \leq 5$ and tails of distributions are observed for $C > 5$. Moreover, distributions of the pixels in the segmentations outputs of Normalized Cuts algorithm shown in Figure 5.8 are not uniform for $2 \leq C \leq 100$.

Another assumption of RI and the suggested Segmentation Fusion algorithm is that distributions computed in the segmentation outputs of two different segmentation algorithms with the same C values are the same. However, this assumption may be invalid as can be observed in this example. In order to analyze this assumption, Kullback-Leibler (K-L) divergence values between the distributions of the segmentation outputs of k-means and Normalized Cut algorithms implemented with the same number of segments $C = k$ are computed and shown in Figure 5.9. For instance, K-L divergence value between the distributions in Figure 5.7.b and Figure 5.8.b is 0.12 and the divergence increases as C increases. If the SF algorithms is employed on the segmentation sets in Figure 5.7 and Figure 5.8, then the employment of average SOD (5.1) for segmentation fusion may fail. In order to overcome the challeges of these assumptions, C values which maximize the performance of the fusion algorithm should be selected and new distance measures should be defined as suggested in the following sections.

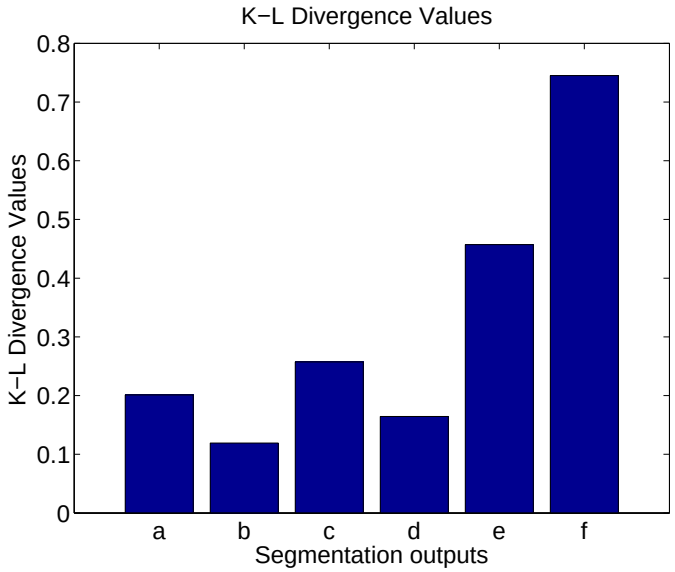


Figure 5.9: Kullback-Leibler(K-L) divergence values between the distributions of the segmentation outputs in Figure 5.7 and Figure 5.8.

5.4 Estimating Number of Different Segment Labels for Segmentation Fusion

One of the crucial problems of Segmentation Fusion algorithm is to estimate, C, the number of different segment labels in the image, as discussed in the previous sections.

In order to estimate C in the base-layer segmentation algorithms, several clustering validity indices can be employed [174]. In this section, a new method is introduced to estimate C for segmentation fusion. For this purpose, a segmentation index (SI) is defined for SF as

$$SI(c) = \sum_{i<j} ARI(s_i, s_j),$$

where $\{s_i\}_{i=1}^{K}$ is the set of K segmentations and each segmentation s_i contains segments with c different labels [165]. Then, the following optimization problem is solved,

$$\hat{C} = \underset{c=2,3,\ldots,C_{max}}{\operatorname{argmax}} \; SI(c) \;, \tag{5.4}$$

where C_{max} is the maximum value of c provided by the user. Therefore, an optimal segment number $\hat{C}$ computed using the base-layer segmentations is the segment number on which all the segmentations agree, i.e. their average agreement quantified with ARI value is maximum.

Vinh and Epps [165] compared Normalized Mutual Information (NMI) and ARI for the estimation of segment number on several datasets. Since they report that both of the algorithms agree on the segment number in various experiments, ARI is employed for estimating c in this work.

5.5 Estimating β Parameter for Segmentation Fusion

β parameter of the Segmentation Fusion (Algorithm 6) controls the convergence rate and performance of the algorithm. For instance, if $\beta = 0$, the algorithm employs pure stochastic BOEM and the algorithm is memoryless [107]. If $\beta = 1$, the algorithm *forgets* slowly [107]. Therefore, if β value is not chosen *appropriately*, then the algorithm may terminate before converging to an *optimal* or a *desired* approximate solution. Zheng et al. experimentally analyzed this challenge for consensus clustering problem [107].

Figure 5.10 depicts the relationship between β values and average ARI values of the algorithms employed in the examples given in Section 5.2 and shown in Figure 5.1, 5.2, 5.3, 5.4, 5.5.

Examples show the random convergence behavior of the Segmentation Fusion (SF) algorithm. For instance, the SF algorithm implemented with $\beta = 0.9$ may converge to a maximum value before a termination time $T = 1000$ is achieved as shown in Figure 5.10.a, 5.10.d and 5.10.e. Moreover, the performance of the SF may decrease after the maximum performance is obtained if the SF continues running until the termination time (see Figure 5.10.b).

On the other hand, the SF algorithm implemented with $\beta = 0.6$ achieves the maximum performance at the termination time $T = 1000$ (see Figure 5.10.b and Figure 5.10.e). In addition, an implementation with $\beta = 0.99$ provides a gradual increase in the performance as the algorithm implemented until $T = 1000$ as shown in Figure 5.10.c.

One of the reasons of the random convergence behavior of the SF algorithm is the stochastic optimization method implemented to compute the best element moves at each iteration; at

each iteration t of the SF, a segmentation is randomly selected according to a uniform distribution. If a segmentation s, which is the best element move, is selected at an iteration t, then a maximum ARI is obtained at that iteration. If the SF continues selecting segmentations in order to obtain a better move, then the performance may decrease since s is the best element move.

Since the SF provides an approximation to the *optimal* segmentation, *the best* segmentation or move is not known during the implementation of the algorithm. Therefore, a new proof strategy is needed to prove the convergence of the stochastic optimization and probabilistic move selection methods of the SF for image segmentation. Since the convergence analysis of the SF considering its computational complexity and approximation error is an open problem, we skip this problem until a future work.

In this work, we intuitively propose an approach to estimate the parameter β by maximizing the agreement between outputs of base-layer segmentation algorithms and an output of the proposed Segmentation Fusion algorithm, following the experimental analyses on the relationship between algorithm performance ARI and parameter β.

Given a set of β values $\Xi = \{\beta_b\}_{b=1}^{B}$, a beta index ($BI$) is defined as

$$BI(\beta_b) = \sum_{i=1}^{K} ARI(s_i, O(\beta_b)),$$

where $O(\beta_b)$ is the output segmentation of the proposed Segmentation Fusion algorithm implemented using β_b. Then the optimal $\hat{\beta}$ is computed by solving the following optimization algorithm:

$$\hat{\beta} = \underset{b=1,2,\ldots,B}{\operatorname{argmax}} \; BI(\beta_b) \; . \tag{5.5}$$

Note that computing $\hat{\beta}$ by solving (5.5) may not provide a *globally optimum* solution for SF. However, we can select a β parameter from the set Ξ which provides a consensus segmentation with the maximum agreement of the base-layer segmentation algorithms that can be achieved by the implementation of SF using the parameters in Ξ.

5.6 Distance Learning

Following the analyses given in the previous sections, a method called distance learning that overcomes the difficulties of the assumptions of SF, such as measuring the distance between two segmentations with different numbers of segments, in this section.

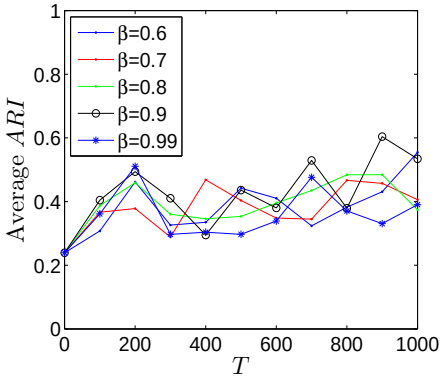
For this purpose, we first analyze the structure of the distance function of SF. Note that the distance function $d(s_i, s_j)$ of SF and a Rand Index ($RI(s_i, s_j)$), that are computed between two segmentations s_i and s_j, are related by the following definition

$$RI(s_i, s_j) = 1 - \frac{d(s_i, s_j)}{\binom{N}{2}},$$

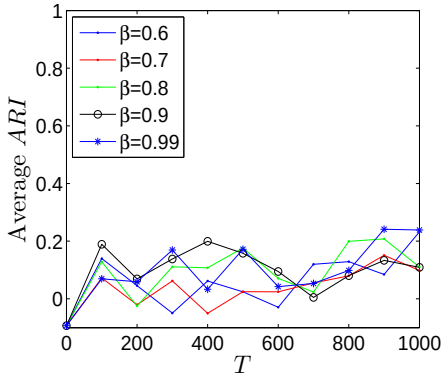
where

$$d(s_i, s_j) = N_{10} + N_{01} \tag{5.6}$$

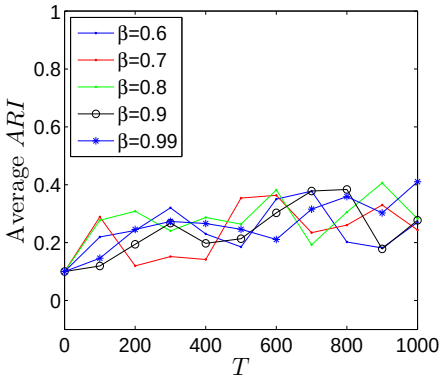
and N_{01} is the number of pairs co-segmented in s_i but not in s_j, and N_{10} is the number of pairs co-segmented in s_j but not in s_i.



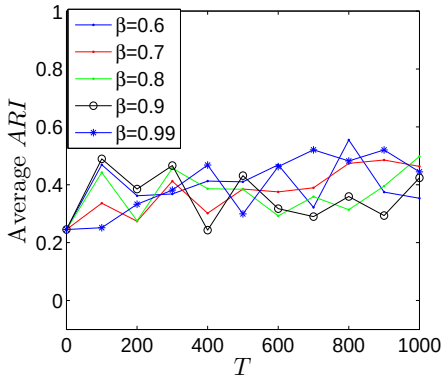
(a) Results for Figure 5.1



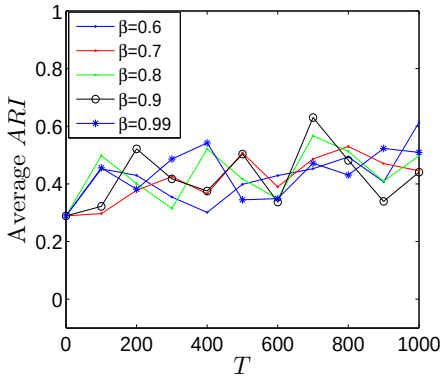
(b) Results for Figure 5.2



(c) Results for Figure 5.3



(d) Results for Figure 5.4



(e) Results for Figure 5.5

Figure 5.10: The relationship between the convergence rate parameter β and Average ARI performance of the Segmentation Fusion algorithm.

Since RI is not corrected for chance, a deficiency of the distance function of SF is observed by analyzing this relationship between $d(s_i, s_j)$ and $RI(s_i, s_j)$. For instance, the average distance between two segmentations is not zero and the distance depends on the number of pixels [19]. Therefore, it is assumed that each segmentation $s_i = \{s_{i,n_i}\}_{n_i=1}^{K_i}$ consists of K_i numbers of different segments in SF.

In order to relax this assumption, we suggest the following distance learning method. Let us first define $\mathfrak{N}(i)$ as the number of pixels in the i^{th} segment of s_i, and $\mathfrak{N}_{ij}$ as the number of pixels in both the i^{th} segment of s_i and the j^{th} segment of s_j. In addition, s_i and s_j are assumed to be randomly drawn with a fixed number of segments, and a fixed number pixels in each segment according to a generalized hypergeometric distribution [173]. Then, an adjusted version of RI called *Adjusted Rand Index* (ARI) [173] is defined as

$$ARI(s_i, s_j) = \frac{\sum_{n_i=1}^{K_i} \sum_{n_j=1}^{K_j} \binom{\mathfrak{N}_{ij}}{2} - a_{ij}}{\frac{1}{2}(a_i + a_j) - a_{ij}} , \tag{5.7}$$

where $a_i = \sum_{n_i=1}^{K_i} \binom{\mathfrak{N}(i)}{2}$, $a_j = \sum_{u_j=1}^{K_j} \binom{\mathfrak{N}(j)}{2}$ and $a_{ij} = \frac{2a_i a_j}{N(N-1)}$.

Note that, if the assumptions are applied for equal segmentation sizes $K_i = K_j, \forall i \neq j$ in ARI, (5.1) is obtained [18]. Instead, $ARI(s_i, s_j)$ is computed for each different base-layer segmentation algorithm output S_i with different segment numbers K_i. Then, $d(s_i, s_j)$ is computed from $ARI(s_i, s_j)$, such that [18]

$$d(s_i, s_j) = 1 - ARI(s_i, s_j).$$

Since $ARI(s_i, s_j)$ values are computed using the segmentation sets for each segmentation pair, the distance function values computed between the segmentation outputs can be *learned* using the segmentation sets independent of the segmentation sizes K_i unlike (5.6).

For instance, let $S_1 = \{s_i\}_{i=1}^{n_1}$ and $S_2 = \{s_i\}_{i=1}^{n_2}$ be two segmentation sets with $n_1 \neq n_2$ obtained from the implementations of a k-means and a Mean Shift segmentation algorithm, respectively. Then, distances between S_1 and S_2 can be computed using the proposed method to measure the similarity between segmentation outputs of k-means and Mean Shift algorithms implemented with different number of segments. Therefore, this method is called Distance Learning (DL) for SF in which $d(s_i, s_j)$ is learned by computing $ARI(s_i, s_j)$ using the segmentation sets.

5.7 Quasi-Distance Learning

An important assumption made in the derivation of ARI [175] is that the number of pixels in each segment is the same. However, this assumption may fail in the segmentation of images that contain complex targets, such as airports or harbors.

In order to relax this assumption, a normalization method is employed for Quasi-distance functions, introduced by Luo et al. [176] as

$$nd(s_i, s_j) = \frac{d(s_i, s_j) - d_{min}(s_i, s_j)}{d_{max}(s_i, s_j) - d_{min}(s_i, s_j)} , \tag{5.8}$$

where $d_{min}(s_i, s_j)$ and $d_{max}(s_i, s_j)$ are the minimal and maximal values of $d(s_i, s_j)$.

Luo et al. [176] states that the exact computation of $d_{max}(s_i, s_j)$ for any segmentation distribution is not known in general, and can be achieved in special cases. For instance, if the sizes of the segments that reside in a consensus segmentation s and a base-layer segmentation s_i are the same, then (5.8) becomes (5.7) and maximal values can be computed exactly [176].

Otherwise, several approximations [176] are proposed such as the computation of approximate $\hat{d}_{max}(s, s_i)$ as

$$\hat{d}_{max}(s, s_i) = \frac{1}{2}\big(\mathcal{C}(s_i, s) + \mathcal{C}(s, s_i)\big) + \big(\mathbb{H}(s_i)\mathcal{C}(s, s_i)\big), \tag{5.9}$$

where $\mathbb{H}(s_i)$ is the entropy of a base-layer segmentation s_i defined as [176]

$$\mathbb{H}(s_i) = \sum_{m=1}^{n_i} p_m \log_2 p_m$$

and $\mathcal{C}(s, s_i)$ is the conditional entropy defined as [176]

$$\mathcal{C}(s, s_i) = \sum_{m=1}^{n_i} \frac{|\pi_m|}{|\pi|}\mathbb{H}(s_{\pi_m}),$$

where $s = \{\pi_1, \pi_2, \ldots, \pi_{n_i}\}$ and s_{π_m} is a sub-partition which contains a segment π_m that co-exist in s and s_i.

In this approximation, distance function between a target segmentation s and a base-layer segmentation s_i is computed as

$$d(s, s_i) = \mathcal{C}(s, s_i) + \mathcal{C}(s_i, s) \tag{5.10}$$

and $d_{min}(s, s_i) = 2\mathcal{C}(s, s_i)$ is its minimal value [176].

An important difference between (5.7) and (5.8) is that the minimal and maximal values of the distances between the pairwise segmentations (s_i, s_j) are included into the distance function as the normalization factors, in order to compute the distances between s_i and s_j in (5.8). On the other hand, (5.7) requires the expected values of the distances between all of the segmentations in the computations. In addition, (5.7) assumes that the number of pixels in the segments are the same, i.e. distributions of the pixels in the segments are the same in different segmentations. However, (5.8) does not make an assumption on the distributions of pixels by computing the distributions using segmentation sets.

In this section, (5.8) is employed to develop another algorithm for distance learning called Quasi-distance Learning(QD). Given a training dataset of segmentations, the algorithm steps can be summarized as follows;

1. Compute $d(s, s_i)$ using (5.10) for a selected segmentation output s_i, $i = 1, 2, \ldots, K$ and a consensus segmentation s computed using a Best One Element Move of SF.
2. Compute the minimal value $d_{min}(s, s_i)$ of $d(s, s_i)$ as $d_{min}(s, s_i) = 2\mathcal{C}(s, s_i)$.
3. Compute the maximal value $d_{max}(s, s_i)$ of $d(s, s_i)$ using (5.9).
4. Compute the values of the quasi-distance function using (5.8).

Then, steps of the Quasi-distance Learning algorithm have been used in the Segmentation Fusion (SF) algorithm for the computations of best one element move at each step of SF.

Note that DL and QD methods are proposed to obtain a consensus segmentation using SF by maximizing the *agreement*, i.e. the similarity between base-layer segmentations and the computed consensus segmentation. In these methods, we do not make any assumption on the availability of a *target* segmentation that is given in a training segmentation set.

On the other hand, one may require to obtain a segmentation output which also agrees with a *target* segmentation, i.e. which is closer to both base-layer segmentations and a *target* segmentation. In order to obtain such a segmentation output of SF, we suggest another distance learning method in the following section.

5.8 Segmentation Fusion with Decision Weighting

The distance learning method proposed in the previous section can be employed for learning *the structure of the distance* in Section 5.6 without using a target segmentation. In this section, distance learning is used for learning the weights w_i assigned to each segmentation s_i in the following fusion problem,

$$\underset{w_i}{\operatorname{argmin}} \sum_{i=1}^{K} w_i d(s_i, s_{target}) \ , \tag{5.11}$$

where s_{target} is the target segmentation, $w_i > 0$ and $\sum_{i=1}^{K} w_i = 1$.

The weights computed in (5.11) provide a fused segmentation output which is closer to a target segmentation s_{target}. If the distance between a base-layer segmentation s_i and s_{target} is *large*, then a weight $w_i \approx 0$ can be assigned to s_i and s_i may be used in the fusion with a confidence less than the other base-layer segmentations. On the other hand, if $d(s_i, s_{target}) \approx 0$, then a large weight $w_i \approx 1$ can be assigned to s_i. Therefore, the problem (5.11) is called as a *Segmentation Fusion with Decision Weighting* problem.

In addition, we require that the computed weights provide a fused segmentation which is both closer to s_{target} and a consensus segmentation on which the base-layer segmentations agree. In order to satisfy both of these requirements, another new weighted distance learning algorithm called Segmentation Weighting is suggested below.

1. First a cumulative Adjusted Rand Index between each base-layer segmentations s_i and s_j is computed as
$$ARI(s_i) = \sum_{j=1}^{K} ARI(s_i, s_j).$$

2. Next, the segmentation weight w_i is computed as follows
$$w_i = \frac{ARI(s_i)}{\sum_{i=1}^{K} ARI(s_i)},$$
where $w_i > 0$ and $\sum_{i=1}^{K} w_i = 1$ for each s_i. Note that w_i is computed according to the agreement between each s_i and all the other base-layer segmentations at this step.

3. Then, w_i is used to compute the weighted distances $w_i d(s_i, s_{target})$, $\forall i = 1, 2, \ldots, K$ in (5.11). Therefore, the agreement between s_i and the base-layer segmentations is used to measure the agreement between s_i and s_{target} in this step.

4. Finally, (5.11) is solved using the suggested Segmentation Fusion Algorithm 6 in the computation of best one element moves from the base-layer segmentations to the target segmentation s_{target}.

A similar partition (i.e. segmentation) weighting method is employed by Yang and Chen [169] using Modified Huber's index, Dunn's Validity Index and Normalized Mutual Information for clustering. In the proposed method, ARI is used instead of these indices as well as other clustering indices, since ARI is directly related to SDD and SOD of SF as mentioned in the previous sections and in [18].

In the training phase, SF algorithm *learns* the weights which are used to produce a consensus segmentation that is both close to a target segmentation and the base-layer segmentations. Then, the *learned* weights are employed on a new segmentation to predict its segment labels. However, one must assure that the statistical properties of training and test data are equivalent in order to employ the learning methods. Note that this requirement may not be satisfied in remote sensing datasets in the experiments because of the variability of the images in the context of space and time, i.e. because of the data shift and domain shift problems [20, 25, 24, 22].

In addition, there are various open problems of the proposed methods. First, there is a trade-off between achieving a consensus segmentation which is obtained by maximizing the agreement between base-layer segmentations and computing a segmentation which is close to a target segmentation. In practical applications, this requirement may not be achieved. For instance, an object which is represented by a target segmentation may not be represented in a set of base-layer segmentations. In addition, the size of a training set of segmentations affects both the performance of the SF and the computation of *optimal* weights. A detailed theoretical analysis of the relationships between the size of the training dataset, learning error, algorithm performances and the trade-off is beyond the scope of this thesis and considered as a future work.

5.9 Incorporating Prior and Side Information to Segmentation Fusion

In this section, we introduce a new Semi-supervised Segmentation Fusion algorithm which solves distance and decision weight learning problems that are mentioned in the previous sections by incorporating side-information about the pixel memberships into the unsupervised Segmentation Fusion algorithm. Then, the goal of the proposed Semi-supervised Segmentation Fusion algorithm can be summarized as obtaining a segmentation which is close to both base-layer segmentations and a target segmentation using weighted distance learning and semi-supervision.

The weights computed in Segmentation Weighting method are mostly greater than zero. However, some of the weights may be required to be zero, in other words, sparsity may be required in the space of weight vectors to select the decision of some of the segmentation algorithms. For instance, if fusion is employed on multi-spectral images with large number of bands, and if some of the most informative bands are needed to be selected, then sparsity defined by the weight vectors becomes a very important property.

In addition, side information about the pixel-wise relationships of the segmentations can be defined in distance functions. Thereby, both the structure of the distance function, the pixel-

wise relationships and the *contributions* of the decisions of the segmentation algorithms can be learned from the data.

5.9.1 Formalizing Semi Supervision for Segmentation Fusion

We define Semi-supervised Segmentation Fusion problem as a convex constrained stochastic sparse optimization problem.

In the construction of the problem, first pixel-wise segment memberships are encoded in the definition of a semi-supervised weighted distance learning problem by decomposing Symmetric Distance Function (SDD) as [103]

$$d(s_i, s_j) = \sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}(s_i, s_j), \tag{5.12}$$

and

$$d_{m,l}(s_i, s_j) = \begin{cases} 1, & \text{if } (m,l) \in \Theta_c(s_i) \text{ and } (m,l) \notin \Theta_c(s_j) \\ 1, & \text{if } (m,l) \notin \Theta_c(s_i) \text{ and } (m,l) \in \Theta_c(s_j) \\ 0, & \text{otherwise} \end{cases},$$

where $(m,l) \in \Theta_c(s_i)$ means that the pixels m and l belong to the same segment Θ_c in s_i and $(m,l) \notin \Theta_c(s_i)$ means that m and l belong to different segments in s_i. Then, a connectivity matrix M is defined with the following elements;

$$M_{ml}(s_i) = \begin{cases} 1, & \text{if } (m,l) \in \Theta_c(s_i) \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

Note that [103],

$$d_{m,l}(s_i, s_j) = \left[M_{m,l}(s_i) - M_{m,l}(s_j) \right]^2 . \tag{5.14}$$

Then, the distance between the connectivity matrices of two segmentations s and s_i is defined as [109]

$$d_\kappa(M(s), M(s_i)) = \sum_{m=1}^{N} \sum_{l=1}^{N} d_\kappa(M_{m,l}(s), M_{m,l}(s_i)) , \tag{5.15}$$

where d_κ is the Bregman divergence defined as

$$d_\kappa(x, y) = \kappa(x) - \kappa(y) - \nabla\kappa(y)(x - y) ,$$

and $\kappa : \mathbb{R} \to \mathbb{R}$ is a strictly convex function. Since d_κ is defined in (5.14) as Euclidean distance, (5.15) is computed during the construction of best one element moves in SF.

In order to compute the weights of base-layer segmentations during the computation of distance functions, (5.11) is converted to the following quadratic optimization problem;

$$\begin{aligned} \underset{\bar{w}}{\operatorname{argmin}} & \sum_{i=1}^{K} w_i d_\kappa(M(s), M(s_i)) + \lambda_q \parallel \bar{w} \parallel_2^2 \\ \text{s.t.} & \sum_{i=1}^{K} w_i = 1, w_i \geq 0, \forall i = 1, 2, \ldots, K , \end{aligned} \tag{5.16}$$

where $\lambda_q > 0$ is the regularization parameter and $\bar{w} = (w_1, w_2, \ldots, w_K)$ is the weight vector. Since we use $\sum_{i=1}^{K} w_i = 1$ and $w_i \geq 0$ in the constraints of the optimization problem (5.16), we enable the *selection* and *removal* of a base-layer segmentation s_i by assigning zero valued weights $w_i = 0$ to s_i.

Defining the distance function (5.12) in terms of the segment memberships of the pixels (5.13) in (5.14), must-link and cannot-link constraints can be incorporated to the constraints of (5.16) as follows;

$$M_{ml}(s_i) = \begin{cases} 1, & \text{if} (m,l) \in \mathfrak{M} \\ 0, & \text{if} (m,l) \in \mathfrak{C} \end{cases}, \tag{5.17}$$

where $\mathfrak{M}$ is the set of must-link constraints and $\mathfrak{C}$ is the set of cannot-link constraints. Then, the following optimization problem is defined for Semi-supervised Segmentation Fusion

$$\begin{aligned} \underset{M(s)}{\operatorname{argmin}} \sum_{i=1}^{K} d_\kappa(M(s), M(s_i)) + \lambda_q \parallel \bar{w} \parallel_2^2 \\ \text{s.t} \quad M_{ml}(s_i) = 1, \text{if} (m,l) \in \mathfrak{M} \\ M_{ml}(s_i) = 0, \text{if} (m,l) \in \mathfrak{C} \,. \end{aligned} \tag{5.18}$$

Wang, Wang and Li [109] analyze generalized cluster aggregation problem using (5.18) for fixed weights $\bar{w}$ and define the solution set as follows;

1. If $(m,l) \in \mathfrak{M}$ or $(m,l) \in \mathfrak{C}$, then (5.17) is the solution set for (k,l),
2. If $(m,l) \notin \mathfrak{M}$ and $(m,l) \notin \mathfrak{C}$, then $M_{ml}(s_i)$ can be solved by

$$\nabla_\kappa M_{ml}(s_i) = \sum_{i=1}^{K} w_i \nabla_\kappa (M(s_i)).$$

Then, they solve (5.16) for fixed $M(s)$.

Note that, ℓ_2 norm regularization does not assure sparsity efficiently [177] because $\parallel \bar{w} \parallel_2^2$ is a quadratic function of the weight variables w_i which treats each w_i equally. In order to control the sparsity of the weights by treating each w_i different from the other weight variables $w_{j \neq i}$ using a linear function of w_i, such as $\parallel \bar{w} \parallel_1$ which is the ℓ_1 norm of $\bar{w}$, a new optimization problem is defined as follows;

$$\begin{aligned} \underset{(M(s),\bar{w})}{\operatorname{argmin}} \sum_{i=1}^{K} w_i d_\kappa(M(s), M(s_i)) + \lambda \parallel \bar{w} \parallel_1 \\ \text{s.t} \quad \sum_{i=1}^{K} w_i = 1, w_i \geq 0, \forall i = 1, 2, \ldots, K \\ M_{ml}(s_i) = 1, \text{if} (m,l) \in \mathfrak{M} \\ M_{ml}(s_i) = 0, \text{if} (m,l) \in \mathfrak{C} \,, \end{aligned} \tag{5.19}$$

where $\lambda \in \mathbb{R}$ is the parameter which defines the sparsity of $\bar{w}$. Similarly, (5.19) is computed into two parts;

1. For fixed $M(s)$, solve

$$\begin{aligned} \underset{\bar{w}}{\operatorname{argmin}} \sum_{i=1}^{K} w_i d_\kappa(M(s), M(s_i)) + \lambda \parallel \bar{w} \parallel_1 \\ \text{s.t} \quad \sum_{i=1}^{K} w_i = 1, w_i \geq 0, \forall i = 1, 2, \ldots, K \ , \end{aligned} \tag{5.20}$$

2. For fixed $\bar{w}$, solve

$$\begin{aligned} \underset{M(s)}{\operatorname{argmin}} \sum_{i=1}^{K} w_i d_\kappa(M(s), M(s_i)) + \lambda \parallel \bar{w} \parallel_1 \\ \text{s.t} \quad M_{ml}(s_i) = 1, \text{if}(m,l) \in \mathfrak{M} \\ M_{ml}(s_i) = 0, \text{if}(m,l) \in \mathfrak{C} \ , \end{aligned} \tag{5.21}$$

An algorithmic description of Semi-supervised Segmentation Fusion which solves (5.20) and (5.21) in Segmentation Fusion algorithm is given in the next subsection.

5.9.2 Semi-supervised Segmentation Fusion Algorithm

In the proposed Semi-supervised Segmentation Fusion Algorithm, (5.20) and (5.21) are solved to compute weighted distance functions which are used in the construction of best one element moves of Segmentation Fusion.

In Algorithm 7, first the weight vector $\bar{w}$ is computed by solving (5.20) for each selected segmentation $s_{i'}$ in the 4^{th} step of the algorithm. In order to solve (5.20) using an optimization method called Alternating Direction Method of Multipliers (ADMM) [112]. Details of the employment of ADMM to solve (5.20) are given in the next subsection.

Once the weight vector $\bar{w}$ is computed in the 4^{th} step, (5.21) is solved in the 5^{th}, 6^{th} and 7^{th} steps of the algorithm: $\bar{w}d(s_{i'}, s) + \lambda \parallel \bar{w} \parallel_1$ is computed using $M(s_{i'})$ and $\bar{w}$ in the 5^{th} step, $[H_t]$ is computed in the 6^{th} step and Δs is computed in the 7^{th} step to update s. Note that the sparse weighted distance function, which is approximated by $\beta[H_t] + [\bar{w}d(s_{i'}, s) + \lambda \parallel \bar{w} \parallel_1]$ in Algorithm 7, is different from the distance function in Algorithm 6.

In addition, each segmentation is selected sequentially in a pseudo-randomized permutation order in Algorithm 7. If an initially selected segmentation performs better than the other segmentations, then the algorithm may be terminated in the first running over the permutation set. Otherwise, the algorithm runs until the termination time T is achieved or all of the segmentations are selected.

input : Input image I, $\{SA_j\}_{j=1}^{J}$, T, T_τ.
output: Output segmentation O.
Run SA_j on I to obtain $S_j = \{s_i\}_{i=1}^{u_j}$, $\forall j = 1, 2, \ldots, J$.;
At $t = 1$, initialize s and $[H_t]$;
for $t \leftarrow 2$ **to** T **do**
    Randomly select one of the segmentation results with an index $i' \in \{1, 2, \ldots, K\}$;
    Solve (5.20) for $M(s_{i'})$ to compute w_k;
    Compute $\bar{w}d(s_{i'}, s) + \lambda \parallel \bar{w} \parallel_1$;
    $[H_t] \leftarrow \beta[H_t] + [\bar{w}d(s_{i'}, s) + \lambda \parallel \bar{w} \parallel_1]$;
    Compute Δs by solving $\underset{n,c}{\operatorname{argmin}} \beta[H_t]_{n,c}$;
    $s \leftarrow s + \Delta s$;
    $t \leftarrow t + 1$;
end
$O \leftarrow s$;

Algorithm 7: Semi-supervised Segmentation Fusion.

5.9.2.1 Computation of the weights of Semi-supervised Segmentation Fusion Algorithm

In order to compute weight vectors assigned to the decisions of base-layer segmentation algorithms in Algorithm 7, (5.20) is re-written in an ADMM problem [112] as

$$\begin{aligned} &\text{minimize } A(\bar{w}) + B(\bar{z}) \\ &\text{s.t} \quad \bar{w} - \bar{z} = 0 \,, \end{aligned} \tag{5.22}$$

where $A(\bar{w}) = \sum_{i=1}^{K} w_i d_\kappa(M(s), M(s_i)$ and $B(\bar{z}) = \lambda \parallel \bar{z} \parallel_1$.

The Lagrangian of (5.22) is

$$L_\theta(\bar{w}, \bar{z}, \bar{\alpha}) = A(\bar{w}) + B(\bar{z}) + \bar{\alpha}^\top(\bar{w} - \bar{z}) + (\frac{\theta}{2}) \parallel \bar{w} - \bar{z} \parallel_2^2, \tag{5.23}$$

where $\theta > 0$ is a penalty parameter, where $\bar{\alpha}$ is a vector of optimization dual variables, $\bar{\alpha}^\top$ is the transpose of $\bar{\alpha}$, and $\| \cdot \|_2^2$ is the squared Euclidean norm.

Then, (5.23) can be solved in the following three steps using ADMM [112];

- $\bar{w}$ minimization step of (5.23):

$$\bar{w}^{\tau+1} := \underset{\bar{w}}{\operatorname{argmin}} L_\theta(\bar{w}, \bar{z}^\tau, \bar{\alpha}_\tau), \tag{5.24}$$

  where $\bar{\alpha}_\tau$ is the dual optimization vector of the Lagrangian of (5.22) computed at an iteration τ.

- $\bar{z}$ minimization step of (5.23):

$$\bar{z}^{\tau+1} := \underset{\bar{z}}{\operatorname{argmin}} L_\theta(\bar{w}^{\tau+1}, \bar{z}, \bar{\alpha}_\tau), \tag{5.25}$$

- $\bar{\alpha}$ update step of (5.23):

$$\bar{\alpha}_{\tau+1} := \bar{\alpha}_{\tau} + \theta(\bar{w}^{\tau+1} - \bar{z}^{\tau+1}). \tag{5.26}$$

In (5.24), the Lagrangian is minimized with respect to $\bar{w}$ at an iteration of the ADMM steps, τ. Then, $\bar{w}$ is updated and used in the second step (5.25) which minimizes the Lagrangian with respect to $\bar{z}$. Finally, the dual optimization variable $\bar{\alpha}$ is updated.

In order to clarify the relationship among the ADMM steps, first, we define the residual function $\bar{r} = \bar{w} - \bar{z}$ such that

$$\bar{\alpha}^{\top}\bar{r} + \frac{\theta}{2} \parallel \bar{r} \parallel_2^2 = \frac{\theta}{2} \parallel \bar{r} + \bar{u} \parallel_2^2 - \frac{\theta}{2} \parallel \bar{u} \parallel_2^2,$$

where $\bar{u} = \frac{1}{\theta}\bar{\alpha}$ is called the scaled dual optimization variable, which is used in ADMM as

$$\bar{w}^{\tau+1} := \underset{\bar{w}}{\operatorname{argmin}}\Big(A(\bar{w}) + \frac{\theta}{2} \parallel \bar{w} - \bar{z}^{\tau} + \bar{u}^{\tau} \parallel_2^2 \Big), \tag{5.27}$$

$$\bar{z}^{\tau+1} := \underset{\bar{w}}{\operatorname{argmin}}\Big(B(\bar{z}) + \frac{\theta}{2} \parallel \bar{w}^{\tau+1} - \bar{z} + \bar{u}^{\tau} \parallel_2^2 \Big), \tag{5.28}$$

$$\bar{u}^{\tau+1} := \bar{u}^{\tau} + \bar{w}^{\tau+1} - \bar{z}^{\tau+1}. \tag{5.29}$$

Then, $\bar{u}$ relates the dual optimization variable $\bar{\alpha}$ to the residual $\bar{r}$ [112] as follows;

$$\bar{u}^{\tau} = \bar{u}^{0} + \sum_{\nu=1}^{\tau} \bar{r}^{\nu},$$

where $\bar{r}^{\nu} = \bar{w}^{\nu} - \bar{z}^{\nu}$ is the residual computed at the iteration ν.

Then, the ADMM algorithm [112] solves (5.20) using the scaled dual optimization variable in the following three steps;

1. $\bar{w}$ update step:

$$\bar{w}^{\tau+1} := \underset{\bar{w}}{\operatorname{argmin}}\Big(A(\bar{w}) + (\theta/2)\|\bar{w} - \bar{z}^{\tau} + \bar{u}^{\tau}\|_2^2\Big), \tag{5.30}$$

   where $\bar{u}^{\tau} = \frac{1}{\theta}\bar{\alpha}_{\tau}$ is a vector of scaled dual optimization variables $\bar{\alpha}_{\tau}$ of the Lagrangian of (5.22) computed at an iteration τ.

2. $\bar{z}$ update step:

$$\bar{z}^{\tau+1} := \Pi_{\frac{\lambda}{\theta}}\Big(\bar{w}^{\tau+1} + \bar{u}^{\tau}\Big), \tag{5.31}$$

   where $\Pi_{\frac{\lambda}{\theta}}\Big(\bar{w}^{\tau+1} + \bar{u}^{\tau}\Big)$ is the soft thresholding operator defined as

$$\Pi_{\frac{\lambda}{\theta}}\Big(\bar{w}^{\tau+1} + \bar{u}^{\tau}\Big) = \begin{cases} \bar{w}^{\tau+1} + \bar{u}^{\tau} - \frac{\lambda}{\theta}, & \text{if } \bar{w}^{\tau+1} + \bar{u}^{\tau} > \frac{\lambda}{\theta} \\ 0, & \text{if } |\bar{w}^{\tau+1} + \bar{u}^{\tau}| \leq \frac{\lambda}{\theta} \\ \bar{w}^{\tau+1} + \bar{u}^{\tau} + \frac{\lambda}{\theta}, & \text{if } \bar{w}^{\tau+1} + \bar{u}^{\tau} < -\frac{\lambda}{\theta} \end{cases} \tag{5.32}$$

   The soft thresholding operator is employed to solve (5.28), since $B(\bar{z})$ is not differentiable.

3. $\bar{u}$ update step:

$$\bar{u}^{\tau+1} := \bar{u}^{\tau} + \bar{w}^{\tau+1} - \bar{z}^{\tau+1}. \tag{5.33}$$

These three steps of ADMM are iterated until a termination criterion $\tau \leq T_{\tau}$ is achieved. Note that, ADMM algorithm converges to an *optimal* solution if i) an *optimal* solution, i.e. weight vector, exists and can be computed using the training data, and ii) there is no gap between primal and dual solutions.

However, we may not obtain an approximate segmentation which is close to an *optimal* segmentation using SSSF, even if we compute an optimal weight vector using ADMM. Because, SSSF implements the stochastic optimization algorithm SF in the computation of best one element moves and the theoretical analysis of the convergence rate of SF is an open question. In this work, we provide an experimental analysis of the convergence of SF and SSSF in the next chapter.

On the other hand, if SSSF converges to an optimal solution given *sufficiently large* T and T_{τ}, then we can achieve the maximum performance, i.e. $ARI_{tr}(s, s_{target} = 1$ computed in the training phase. However, we may suffer from a generalization error $ARI(\hat{s}, s_{test}$ which is defined as the error occurred in the prediction of a segmentation output $\hat{s}$ given a test segmentation s_{test}. Therefore, another fascinating challenge is the analysis of the relationship between T, T_{τ}, $ARI_{tr}(s, s_{target})$ and $ARI(\hat{s}, s_{test})$, which is considered as the future work.

5.10 Computational Complexity of the Algorithms

The motivation for the development of Segmentation Fusion algorithm is to reduce the computational complexity of the the median segmentation computation problem for image segmentation, which is an NP-complete problem [18]. Algorithms that provide approximate solutions to the problem have polynomial complexity in the number segments N or the number of segmentations K [18]. For instance, Best of K (BOK) algorithm has the computational complexity $O(K^2N)$ and is a 2-approximation algorithm, i.e. the sizes of the segments are close to the optimal segment sizes by a factor of 2 in the fused segmentation. In addition, the computational complexity of the well known clustering based approaches, such as majority voting of the segmentation outputs, is dominated by the complexity $O(N^2K)$ of the initialization step in which the distance matrices are computed explicitly [18].

In the proposed SF algorithm, BOK algorithm is implemented with complexity $O(K^2)$, where $N = 1$ since the best partition is selected for the initialization step. Each update step of the SF takes CN SDD operations. If $[H_{t=0}]$ is need to be computed accurately at initialization, $KJCN$ extra SDD evaluations are needed. Therefore, the computational complexity of SF is $O(CN(K + T))$, if the algorithm terminates after T iterations. If $[H_{t=0}]$ is initialized as an all-zero matrix, then the computational complexity is reduced to $O(CNT)$.

Computation of ARI and normalized indices takes $O(C^2N)$ time. Since SF employs a stochastic optimization procedure in which a single sample is selected at each iteration, complexity of the computation of the distance functions is $O(CN)$ in SF. Therefore, computational complexity of distance learning algorithms proposed in this chapter has the same complexity with SF, which is $O(CNT)$, if the SF terminates after T iterations. However, Segmentation Weighting requires the computation of ARI values for each segmentation pair. Therefore, the computational complexity of the Segmentation Weighting algorithm is $O(C^2NT)$.

Estimation of the parameters C and β requires running the SF C_{max} and B times, therefore, their computational complexities are $O(CNTC_{max})$ and $O(CNTB)$, respectively. However, the estimation procedures can be run in different processes in parallel. If they are implemented in Z parallel processors, then the complexities are $O(\frac{CNTC_{max}}{Z})$ and $O(\frac{CNTB}{Z})$, respectively.

In Semi-supervised Segmentation Fusion, the weight vectors are computed by solving (5.20) using an optimization algorithm called ADMM, which takes $O(NT_\tau)$ [112], if ADMM terminates after T_τ iterations. Then, the segmentation s is updated using the steps in Segmentation Fusion which takes $O(CN)$ for each iteration $t \leq T$. Therefore, the computational complexity of SSSF is $O(N^2TCT_\tau)$).

5.11 Chapter Summary

Segmentation problem has been analyzed for unsupervised, supervised and semi-supervised learning. In unsupervised segmentation problem, target labels of a segmentation, i.e. a target segmentation is not available. Therefore, domain specific heuristics may be employed to compute an error function to measure the similarity between segments in a segmentation obtained from a segmentation algorithm.

One of the approaches employed to analyze the similarity between different segmentations is the computation of a consensus segmentation by maximizing the agreement between decisions of base-layer segmentation algorithms. In order to solve the consensus segmentation problem, we have proposed a Segmentation Fusion (SF) algorithm which selects and/or fuses the segmentation outputs of several segmentation algorithms in order to achieve a consensus of base-layer segmentations. The output segmentation of the SF algorithm can be interpreted as representation of segmentation with maximum mutual information on the set of base-layer segmentations.

SF is examined by employing two popular segmentation algorithms at the base-layer and applied on remote sensing applications: Mean Shift and RHSEG. Images consisting of several complex objects, such as substations, airports and harbours have been analyzed. The consensus output demonstrates its efficacy in compromising over-segmented results and under-segmented results and enhance the performance of the algorithm without preselecting parameters.

Various challenges of Segmentation Fusion (SF) such as parameter estimation and distance learning are addressed to overcome the difficulties of the assumptions of SF. First, a new method is proposed to estimate the number of different segment labels C in the images for the suggested SF, and a method is introduced to estimate the convergence rate parameter β of the SF.

In order to estimate the algorithm parameter β, an index called beta index is used by measuring the agreement between an output of a base-layer segmentation algorithm and an output of the Segmentation Fusion algorithm that is computed using a parameter value $\beta_b \in \Xi$, $i = 1, 2, \ldots, B$. An index, called segmentation index, which measures the similarity between the segmentations computed using different number of segment labels $c = 1, 2, \ldots, C$, is introduced to select the *optimal* C.

Second, two novel methods are proposed to learn the distance function employed in the SF

using prior information about the pixels, segments and segmentations, such as the distributions of pixels or the number of segments in segmentations.

The distance functions are learned by analyzing the statistical properties of segmentations in order to enhance the segmentation performance without preselecting parameters, or evaluating the outputs for specific targets. For this purpose, two approaches have been suggested. First distance learning algorithms are embedded into the Segmentation Fusion algorithm using two suggested methods called Distance Learning for SF (DL) and Quasi-distance Learning (QD).

In supervised segmentation problem, image segmentation is employed as a preprocessing step for target detection and recognition in remote sensing applications, such as for building detection given in Section 4.1. Therefore, extracting the segments that represent targets of interest is crucial. For instance, the region representing a building should be extracted as a whole segment in order to extract the texture features that contain information on building roofs. Then, the classification algorithms are trained by a set of discriminative features, extracted from each segment. Another example is to detect ships and jetties in harbors and runways in airports, using the shape information extracted from the boundaries of the regions.

Different segmentation algorithms with different parameters provide different segmentation results, which construct a set of base-layer segmentations. Therefore, the problem can be defined as the selection or the fusion of the segmentation outputs of the base-layer segmentation algorithms in a segmentation fusion problem in order to achieve a fused segmentation which is both close to a target segmentation and base-layer segmentations.

For this purpose, first *contributions* of the base-layer segmentation algorithms to decision fusion are measured as the error values induced by their decisions to the error function of the segmentation fusion problem. Then, the weighted distance learning problem is defined as a weighted decision (i.e. segmentation) fusion problem. To solve this problem using supervision, performance validation indices are employed to compute the weights [169].

Finally, an algorithm called Semi-supervised Segmentation Fusion (SSSF) is introduced for fusing the segmentation outputs (decisions) of base-layer segmentation algorithms by incorporating the prior information about the data statistics and side-information about the content into the Segmentation Fusion algorithm. The proposed SSSF algorithm reformulates the SF algorithm as a constrained optimization problem, where the constraints are defined in such a way to semi-supervise the segmentation process.

Various open problems of the proposed algorithms, such as the analyses of relationships between statistical properties of datasets, convergence rates, training and test performances of the proposed algorithms, are postponed to the future work.

In the next chapter, the proposed algorithms are examined on benchmark datasets which contain multi-spectral [178, 179] and aerial images [180].

CHAPTER 6

EXPERIMENTAL ANALYSES AND APPLICATIONS OF UNSUPERVISED AND SEMI-SUPERVISED SEGMENTATION FUSION ALGORITHMS

In this chapter, the proposed Unsupervised and Semi-supervised methods are analyzed with simulations and their behaviors are examined on real world benchmark multi-spectral and aerial images [178, 179, 180].

In the experiments, four well-known segmentation algorithms, k-means, Mean Shift [145],[144], Normalized Cuts [181] and Graph Cuts [182, 183, 184, 185, 186, 187] are used as the base-layer segmentation algorithms. Mean Shift and k-means are summarized in the previous chapters. Normalized Cuts and Graph Cuts algorithms are briefly described in the following section.

6.1 Graph Partitioning Algorithms for Image Segmentation

In graph partitioning algorithms which solve an *optimal* graph cut problem, an image $\mathbb{I}$ is represented by a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a set of nodes that represent pixels in the image and $\mathcal{E}$ is a set of edges that represent the connections between pixels. The similarity or the relation between two pixels $v_m \in \mathcal{V}$ and $v_l \in \mathcal{V}$ is represented by a weight $w_{ml} \in \mathcal{W}$ of an edge $\mathfrak{e}_{ml} \in \mathcal{E}$. The pixel-wise similarity is extended to segment-wise similarity between to disjoint segments $\mathcal{V}_i \subset \mathcal{V}$ and $\mathcal{V}_j \subset \mathcal{V}$, such that $\mathcal{V}_i \cap \mathcal{V}_j = \varnothing$ and $\mathcal{V}_i \cup \mathcal{V}_j = \mathcal{V}$, by a graph cut

$$cut(\mathcal{V}_i, \mathcal{V}_j) = \sum_{v_m \in \mathcal{V}_i, v_l \in \mathcal{V}_j} w_{ml} \ . \tag{6.1}$$

Since $cut(\mathcal{V}_i, \mathcal{V}_j)$ computes the sum of weights (capacities) between the pixels in $\mathcal{V}_i$ and $\mathcal{V}_j$, the cut can be used as a measure of the similarities between two segments. Then, an *optimal* bi-partitioning of the graph, i.e. an *optimal* segmentation of the image with $C = 2$ different number of segments, can be computed by minimizing (6.1). Therefore, the minimum cut (min-cut) maximizes the dissimilarity between the segments. However, finding a segmentation which minimizes (6.1) is an NP-Hard combinatorial optimization problem. In order to solve this problem in polynomial time, (6.1) should be relaxed by reformulating the min-cut problem as a max-flow (maximum flow) problem [188, 189, 185]. In max-flow formulation, first, two terminal nodes, namely source and sink nodes, which belong to two disjoint sets are defined. Then, defining the flow capacity among the nodes as the weights, the capacity of a minimal

cut between source and sink (minimum s-t cut) can be measured as the maximal amount of a flow from source to sink. Thereby, min-cut problem in planar graphs is reduced minimum s-t cut problem, which can be solved in polynomial time [188, 189, 185]. In the experiments, a Graph Cut implementation of Veksler [186] for image segmentation is used with a Matlab wrapper of Bagon [184].

If a cut minimization algorithm minimizes the number of edges across the cut by solving (6.1), then the solution may provide segments which are too small to represent the objects, such that single pixels can be considered as segments. In order to solve this problem, a constraint can be used in (6.1) to model the relationship between a disjoint partition $\mathcal{V}_i$ and $\mathcal{V}$ as

$$Ncut(\mathcal{V}_i, \mathcal{V}_j) = \frac{cut(\mathcal{V}_i, \mathcal{V}_j)}{assoc(\mathcal{V}_i, \mathcal{V})} + \frac{cut(\mathcal{V}_i, \mathcal{V}_j)}{assoc(\mathcal{V}_j, \mathcal{V})}, \tag{6.2}$$

where

$$assoc(\mathcal{V}_i, \mathcal{V}) = \sum_{v_m \in \mathcal{V}_i, v_l \in \mathcal{V}} w_{ml}, \tag{6.3}$$

which measures the sum of weights of all edges that are connected to the pixels in $\mathcal{V}_i$.

Then, the Normalized Cut problem is defined as minimizing (6.2). Note that solving the Normalized Cut problem on regular grids is an NP-complete problem [190]. However, Shi and Malik suggest a method to obtain an approximate solution to the problem using spectral properties of the Graph [181]. The key idea is to use the second smallest eigenvalue of a *normalized* Graph Laplacian [191] in the computation of (6.2). Then, an approximate solution of (6.2) can be computed in polynomial time. Shi and Malik [181] define the steps of the Recursice 2-way Ncut algorithm as follows;

1. Construct a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ using an image $\mathbb{I}$,

2. Solve $(\mathcal{D} - \mathcal{W})\bar{\iota} = \lambda\mathcal{D}\bar{\iota}$ to compute the eigenvectors with the smallest eigenvalues, where $\mathcal{D} = diag(\mathfrak{d}_1, \ldots, \mathfrak{d}_N)$, $\mathfrak{d}_m = \sum_l w_{ml}$, $N = |\mathcal{V}|$, and $\bar{\iota}$ is an indicator vector such that $\bar{\iota}_m = 1$, if the m^{th} node is in $\mathcal{V}_i$ and $\bar{\iota}_m = -1$, otherwise.

3. Partition the graph into two parts by finding the splitting point at which (6.2) is maximized using the eigenvector with the second smallest eigenvalue obtained from the previous step.

4. If the value of (6.2) is smaller than a user defined threshold τ_{ncut}, then recursively partition the segmented parts. Otherwise, exit.

In the implementations, the source code provided by Shi [181] is used. A weight between the m^{th} node and the l^{th} node is computed as

$$w_{ml} = part_1 * part_2, \tag{6.4}$$

where

$$part_1 = \exp\left(\frac{- \parallel \bar{x}^r(m) - \bar{x}^r(l) \parallel_2^2}{\sigma_r}\right),$$

$$part_2 = \begin{cases} \exp(\frac{-\|\bar{x}^s(m)-\bar{x}^s(l)\|_2^2}{\sigma_s}), & \text{if } \| \bar{x}^s(m) - \bar{x}^s(l) \|_2 \leq r_{ncut} \\ 0, & \text{otherwise} \end{cases},$$

$\bar{x}^r(m)$ is the spatial location of the m^{th} node, $\bar{x}^r(i)$ is the feature vector at that node, σ_r, σ_s and r_{ncut} are user defined parameters.

In the experiments, user defined parameters are selected by first computing ARI values between a given target segmentation and each segmentation computed for each parameter $\sigma_r \in \{0.1, 0.2, \ldots, 10\}$, $\sigma_s \in \{1, 2, \ldots, 100\}$, $r_{ncut} \in \{1, 2, \ldots, 100\}$ and $\tau_{ncut} \in \{0.01, 0.02, \ldots, 1\}$. Then, a parameter tuple $(\sigma_r, \sigma_s, r_{ncut}, \tau_{ncut})$ which maximizes ARI is selected. Similarly, a parameter tuple $(h_s, h_r, minArea)$ which maximizes ARI is selected for Mean Shift algorithm from the parameter sets $h_s \in \{1, 3, 5, 10, 50, 100\}$, $h_r \in \{1, 3, 5, 10, 50, 100\}$ and $minArea \in \{100, 200 \ldots, 10000\}$. For k-means, $k = C$ is used, if not stated otherwise.

6.2 Analyses Using Simulations

One of the essential assumptions of Segmentation Fusion method suggested in this study is that the elements of the segmentation set $\{s_i\}_{i=1}^K$ are the segments with noise added to a *target* consensus segmentation s. The noise is defined as different labellings and segmentations of a randomly selected subset of pixels in a given base-layer segmentation s_i. The noise level nl is defined as the percentage of pixels replaced between the segments in the segmentation s_i according to the distribution of pixels in the segments in a target segmentation s such that the labels of pixels in s are modified to produce s_i [18].

For instance, given a *target* consensus segmentation s with $N = 100$ pixels, a pixel is randomly selected to be moved to a randomly selected segment according to a uniform distribution, excluding its current segment and including a new *empty* segment, in order to construct a segmentation s_i with $nl = 1$ [18]. If a segmentation s_i is constructed with a noise level $nl = 10$ using s, then random selection and movement of pixels between segments is repeated $100\frac{nl}{N} = 10$ times.

Noisy segmentation generation setup [18] used in the simulations is shown in Figure 6.1. First a target segmentation s is constructed by the employment of an image segmentation (I.S.) algorithm on a given input image $\mathbb{I}$. Then, K segmentations are generated by adding noise to s with a noise level nl (S.G.) using the procedure given above. Finally, the segmentation set is fed to the Semi-supervised Segmentation Fusion (SSSF) algorithm to produce an output segmentation.

In the simulations, each segmentation in the set $\{s_i\}_{i=1}^K$ is generated either using same or different noise levels. In the experiments, must-link and cannot-link constraints are not explicitly defined in order to examine both Algorithm 6 and Algorithm 7, since Algorithm 7 simulates Algorithm 6, if semi-supervised constraints are not introduced in Algorithm 7.

In Figure 6.2, k-means algorithm is used for $k = 2, 4, 6, 8, 10$ for the initial segmentation. Then, $K = 2, 4, 6, 8, 10$ number of different segmentations are generated with noise level

$$nl = 10, 20, 30, 40, 50, 60, 70, 80, 100.$$

In order to measure the performance of SSSF, RI and ARI values are computed between

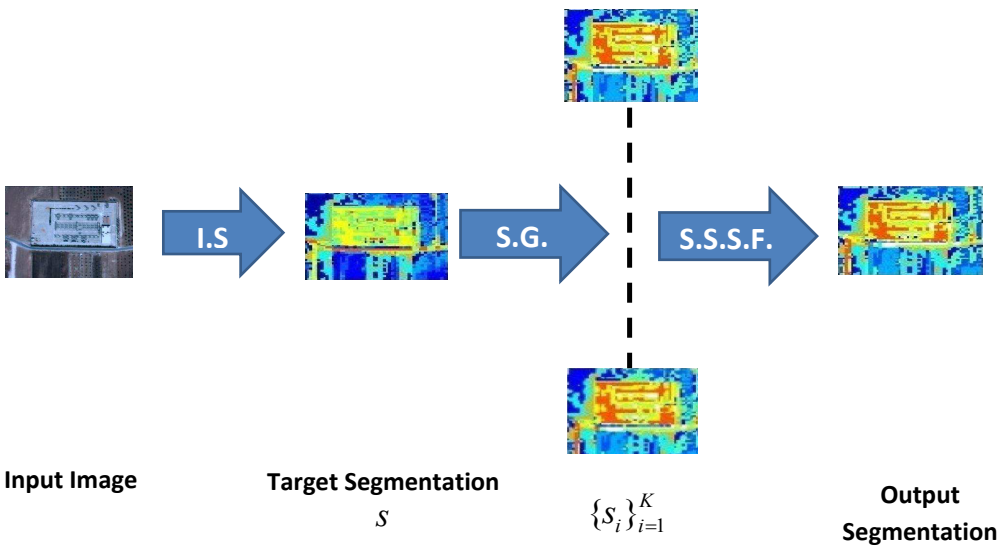
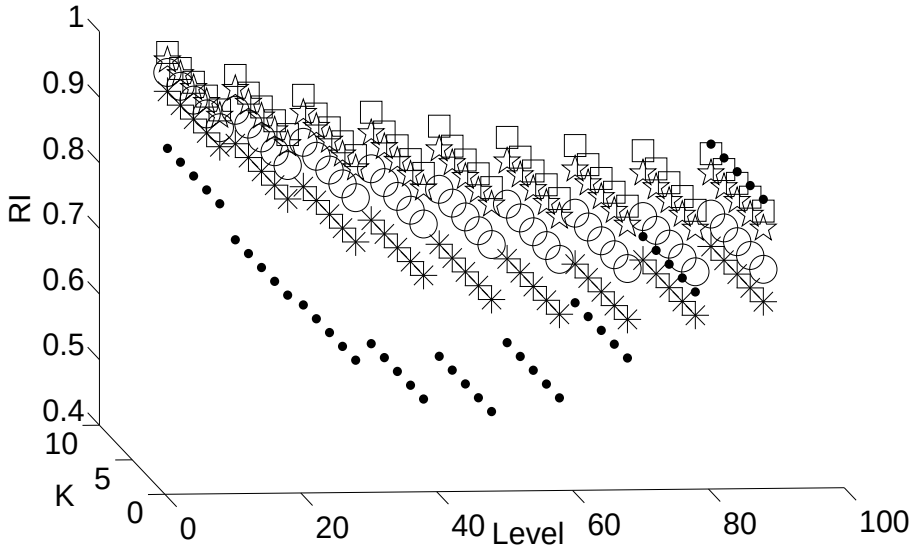


Figure 6.1: The setup for generating segmentation outputs $\{s_i\}_{i=1}^{K}$ with noise, given a noiseless initial segmentation s.

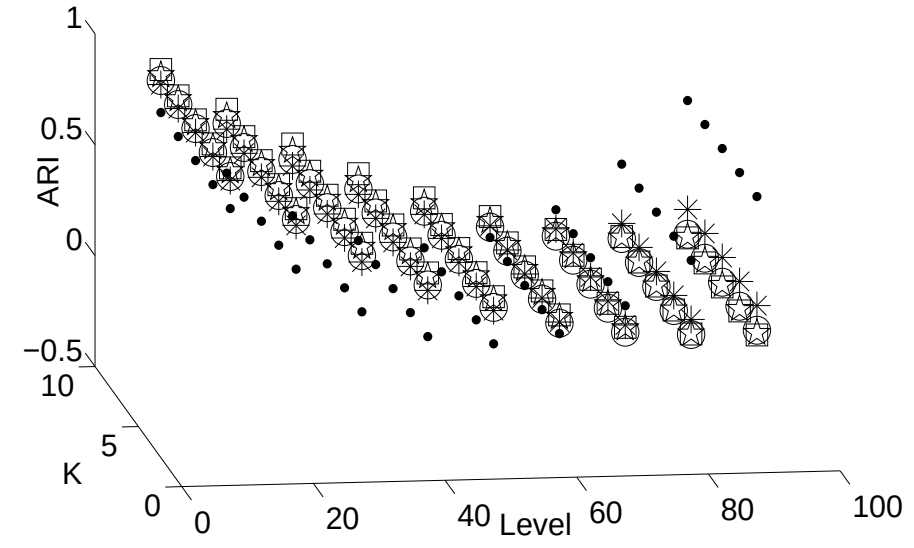
the segmentations of the initial segmentation output of k-means algorithm and the output segmentations which are computed by SSSF algorithm.

In the experimental results, it is observed that as the number of segments C (determined by k) increases for a fixed noise level, RI and ARI decreases. However, a symmetry in the change of RI and ARI values is observed for small number of C, such as $C = 2$, as the noise level increases. Since the noise incorporated to a segmentation s for the noise levels nl and $100 - nl$ are the same for $C = 2$, this symmetry is observed. For instance, there is no noise incorporated to the segmentation s for $nl = 0$, and the labels of all of the pixels are switched from either segment 1 to segment 2 or from segment 2 to segment 1 for $nl = 100$. Therefore, $RI = 0.5$ and $ARI = 0$ at $nl = 50$, and RI and ARI values increase after that point, symmetrically. Since the randomness employed for noise generation increases as k increases, the symmetry disappears as the number of segments k increases. For $k > 6$ and $nl > 50$, ARI values converge to 0 for all of the output segmentations. However, RI values increase for all of the segmentations and k number of segments.

Moreover, RI and ARI values are observed as the same for a fixed noise level nl, the k-means parameter k and different number of segmentations K. In order to observe this phenomenon in detail, the variation of RI and ARI values is shown as K increases for noise level 10 in Figure 6.3.a and Figure 6.3.c, respectively. The changes of RI and ARI values are shown for noise level 50 in Figure 6.3.b and Figure 6.3.d, for $k = 2(\bullet)$, $k = 4(*)$, $k = 6(\bigcirc)$, $k = 8(\star)$, $k = 10(\square)$. It is observed that RI and ARI values are the same as K increases for $k < 10$. However, the difference between RI and ARI values of different segmentations obtained with

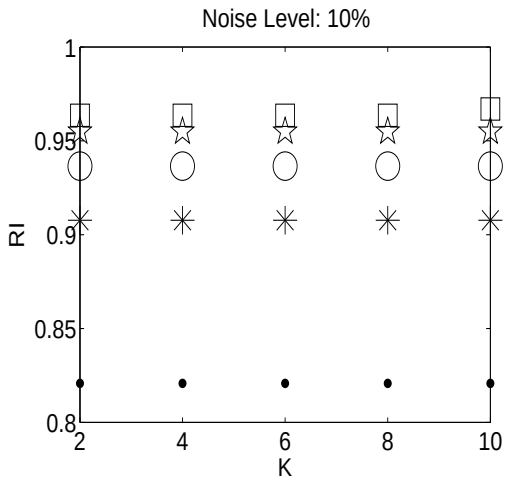


(a) Noise level vs. K vs. *RI*

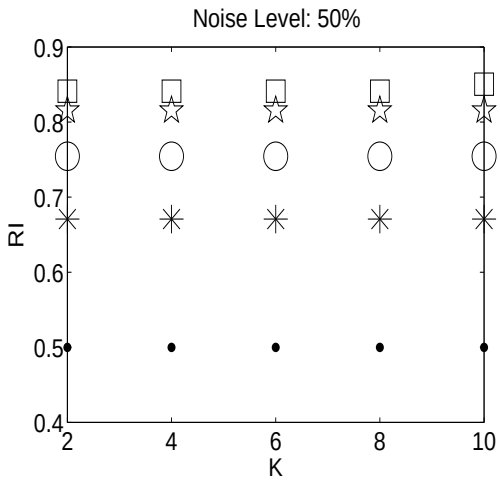


(b) Noise level vs. K vs. *ARI*

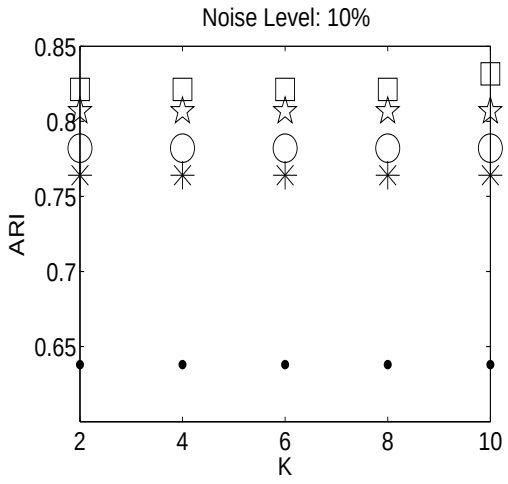
Figure 6.2: Simulation results for $k = 2(\bullet)$, $k = 4(*)$, $k = 6(\bigcirc)$, $k = 8(\star)$ and $k = 10(\square)$.



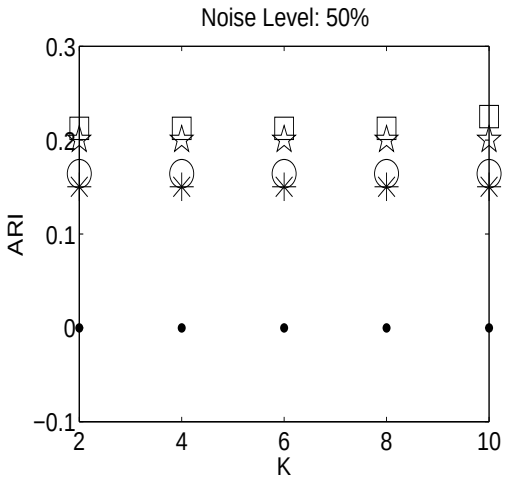
(a) RI values for noise level 10%.



(b) RI values for noise level 50%.



(c) ARI values for noise level 10%.



(d) ARI values for noise level 50%.

Figure 6.3: Change of ARI as K changes for noise levels 10, 50 and for k = 2(•), k = 4($*$), k = 6(◯), k = 8(☆), k = 10(□) .

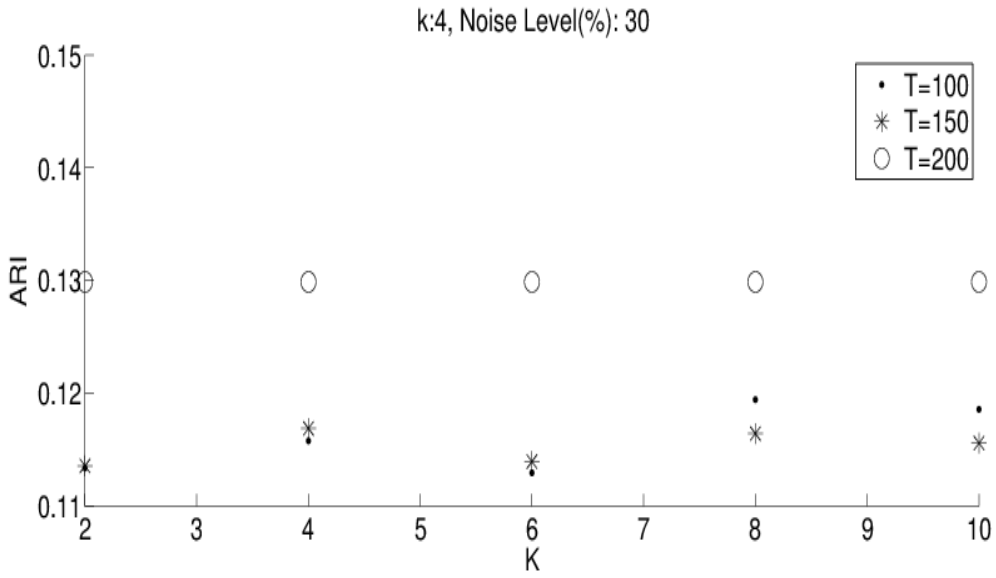


Figure 6.4: The change of performance for the termination time T.

different k values increases as k increases. In addition, the difference between RI and ARI values decreases as noise level increases.

For k = 10, it is observed that the difference between RI and ARI values increases, as K increases. The reason for the fluctuation is that SSSF algorithm is terminated before the consensus is achieved. In order to analyze this observation, the change in the fluctuation pattern is depicted with the change of the termination time T in Figure 6.4, for k = 4 and noise level 30%. Fluctuation for T = 100 and T = 150 is observed, since the algorithm is terminated before the consensus is achieved. When the termination time is increased to T = 200, the same ARI values are obtained for different K, since the algorithm is terminated at t = 184.

6.3 Analyses on Benchmark and Real-world Datasets

In the implementations, k-Means, Normalized Cuts, Graph Cuts and Mean Shift algorithms are used as base-layer segmentation algorithms. Three indices are used to measure the performances between the output images O and the ground truth of the images: i) Rand Index (RI), ii) Adjusted Rand Index (ARI), and iii) Adjusted Mutual Information (AMI) [175] which adjusts the effect of mutual information between segmentations due to chance, similar to the way the ARI corrects the RI. Given two segmentations s_i and s_j, AMI is defined as

$$AMI(s_i, s_j) = \frac{\mathcal{MI}(s_i, s_j) - E\{\mathcal{MI}(s_i, s_j)\}}{max\{\mathbb{H}(s_i), \mathbb{H}(s_j)\} - E\{\mathcal{MI}(s_i, s_j)\}},$$

where $\mathcal{MI}(s_i, s_j)$ is the mutual information between s_i and s_j, $E\{\mathcal{MI}(s_i, s_j)\}$ is the expected mutual information computed over the segments in the segmentations, $\mathbb{H}(s_i)$ and $\mathbb{H}(s_j)$ are the entropies associated to the segmentation sets s_i and s_j.

SSSF is examined using weighted distance functions with semi-supervision. The termination parameter of SSSF and ADMM is taken as T = 1000 and T_τ = 1000, respectively. The penalty

parameter of ADMM is chosen as $\theta = 1$ as suggested in [112]. The regularization parameter is computed as $\lambda = 0.5\lambda_{max}$ [112], where

$$\lambda_{max} = max\{\|d_\kappa(M(s), M(s_1))\mathbf{y}\|_2, \ldots, \|d_\kappa(M(s), M(s_K))\mathbf{y}\|_2\} ,$$

$y_n = \|d_\kappa(M(s), M(s_n))\bar{w}\|_2$, $S = \{s_n\}_{n=1}^N$ is the set of segments in an training image and $\bar{y} = [y_1, y_2, \ldots, y_N]$ is the labels of segments in S. Then, λ is computed in training phase and employed in both training and test phases.

In the training phase, λ and $\bar{w}$ are computed, and the constraints $\mathfrak{M}$ and $\mathfrak{C}$ are constructed using the ground data, i.e. pixel labels of training images as described in Section 5.9.1. In the testing phase, (5.13) is employed for the construction of connectivity matrices and $[\bar{w}d(s_i k, s) + \lambda \parallel \bar{w} \parallel_1]$ is computed $\forall i = 1, 2, \ldots, K$.

6.3.1 Analyses on Multi-spectral Images

In the first set of experiments, the proposed algorithms are employed on 7 band Thematic Mapper Image which is provided by MultiSpec [180]. The image with size 169×169 is splitted into training and test images: i) a subset of the pixels with coordinates $x = (1 : 169)$ and $y = (1 : 90)$ is taken as the training image and ii) a subset of the pixels with coordinates $x = (1 : 169)$ and $y = (91 : 142)$ is taken as the test image. Dataset is split in order to obtain segments with at least 100 pixels both in training and test images.

Training and test images are shown in Figure 6.5.a and Figure 6.5.c, with their ground truth labels in Figure 6.5.b and Figure 6.5.d. In the images, there are $C = 6$ number of different segment labels. The distribution of pixels given the segment labels is shown in Figure 6.6.a and Figure 6.6.b, for training and test datasets, respectively.

First k-means is implemented on different bands $\mathbb{I}_j$ of the multi-spectral image $\mathbb{I} = (\mathbb{I}_1, \mathbb{I}_2, \ldots, \mathbb{I}_J)$ for $J = 7$, in order to perform multi-modal data fusion of different spectral bands using segmentation fusion. The termination time of the SF is set to $T = 1000$. Assuming that C is not known in the image, (5.4) is employed using the training data in order to find the optimal C for $c = 2, 3, 4, 5, 6, 7, 8, 9, 10$. Then, $\hat{C} = 6$ is obtained with $ARI = 0.2648$. (5.5) is employed for a set of β values

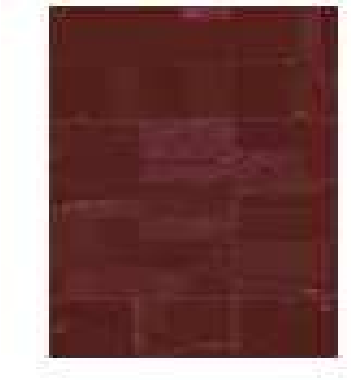
$$\Xi = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$$

and $\hat{\beta} = 0.9$ is obtained with $ARI = 0.2648$.

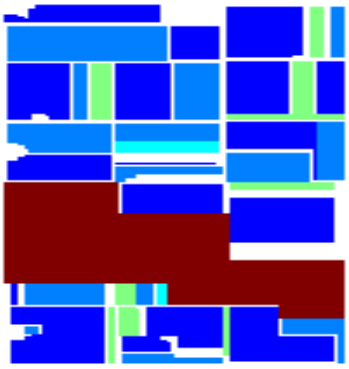
Table 6.1: Training (Tr) and test performances of the algorithms for Thematic Mapper Image.

	Average Base		**SF**		**DL**		**QD**	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.730	0.703	0.731	0.704	0.738	0.710	0.732	0.714
ARI	0.264	0.159	0.265	0.160	0.282	0.184	0.270	0.174
AMI	0.182	0.187	0.182	0.188	0.205	0.203	0.198	0.204

The results of the experiments on Thematic Mapper Image are given in Table 6.1 and Table 6.2. In the **Average Base** column, the performance values of k-means algorithm averaged over



(a) Training image.

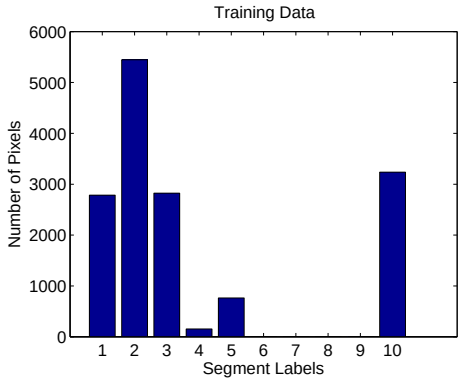


(b) Ground truth of training image.

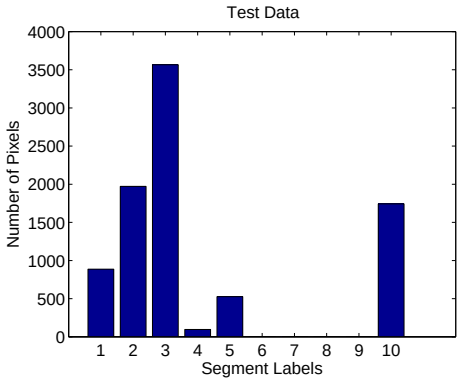
(c) Test image.

(d) Ground truth of test image.

Figure 6.5: Training and test images obtained from Thematic Mapper Image.



(a) Distribution of pixels in training dataset.



(b) Distribution of pixels in test dataset.

Figure 6.6: Distribution of pixels given the segment labels in Thematic Mapper Image. Note that the distributions of test and training data are quite similar.

Table 6.2: Training (Tr) and test performances of the algorithms for Thematic Mapper Image.

	SW		**SSSF**	
	Tr	**Test**	**Tr**	**Test**
RI	0.740	0.718	0.792	0.740
ARI	0.272	0.181	0.305	0.220
AMI	0.197	0.202	0.251	0.237

7 bands are given. The performance values of the Segmentation Fusion algorithm are given in the column labeled **SF**, and that of the Semi-supervised Segmentation Fusion algorithm are given in the column labeled **SSSF**. The performance of Distance Learning and Quasi-distance Learning algorithms, are given in **DL** and **QD**, respectively. It is observed that the performance values of **SF** are similar to the arithmetic average of the performance values of k-means algorithms. Since distance functions for **SF** are computed using the segmentation-wise ARI values in **DL** and **QD**, ARI values of **DL** and **QD** increase compared to **SF**.

The performances of Segmentation Weighting algorithm are given in **SW** (see Table 6.2). In **SW**, the weights of each segmentation are computed as a function of ARI values of the segmentations using the training images and the weighted distance functions are employed for the experiments on test images. Note that the segmentation with higher ARI values have weight values greater than the segmentation with lower ARI. Therefore, the distance values computed by k-means algorithms on the channels of the image using the segmentations with lower ARI values are suppressed by the segmentations with higher ARI. In addition, the output segmentation of **SW** is closer to the segmentation with higher ARI. In the experiments, it is observed that the performance values of **SW** are greater than the performance values of the other algorithms except **SSSF**.

When semi-supervision is used, a remarkable increase is observed in the performances in **SSSF**. However, full performance (1 values for the indices) is not achieved in training. Since the output image O may not converge to the ground truth image, the convergence assumption mentioned in the previous chapter may not be valid for this image.

In the second set of the experiments, k-means, Graph Cut and Mean Shift algorithms are employed on 7-band training and test images. Now, the image segmentation problem is considered as a pixel clustering problem in 7 dimensional spaces. $\hat{C} = 6$ and $\hat{\beta} = 0.9$ are obtained with $ARI = 0.267$ using the training data and with $ARI = 0.176$ for test data.

The results are given in Table 6.3 and Table 6.4. The performance values of **SF** are closer to the performance values of the **Mean Shift** algorithm, since the output image of **SF** is closer to the output segmentation of the **Mean Shift** algorithm. It is observed that the ARI values of **DL** are greater than the values of **QD**, since **DL** computes the distance functions by computing the ARI values between the segmentations. However, the AMI values of **QD** are greater than the values of **DL**, since **QD** calibrates distance functions by computing information theoretic measures, such as conditional entropty of the segmentations, which are related to the mutual information of the segmentations computed in AMI.

Moreover, **SW** and **SSSF** provide better performances than the other algorithms, since **SW** and **SSSF** incorporate prior information by assigning higher weights to the partitions with higher performances.

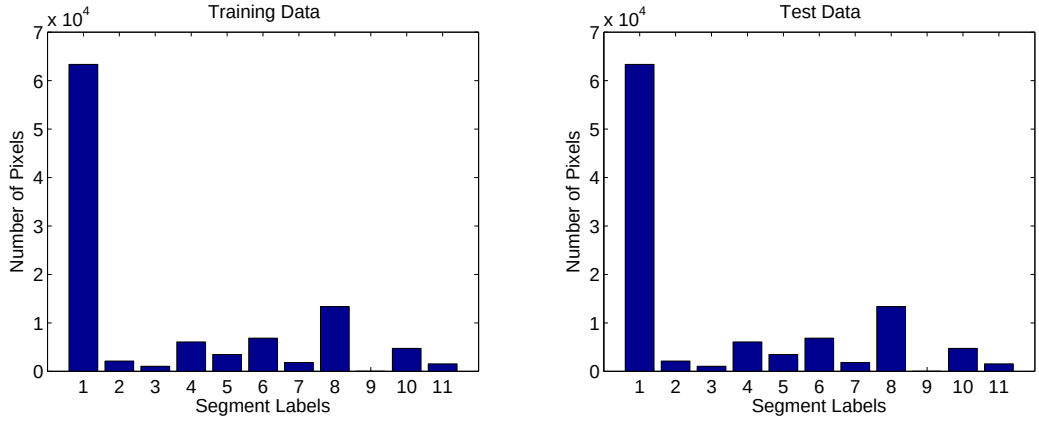
Table 6.3: Experiments using k-means, Graph Cut, Mean Shift and Segmentation Fusion algorithms on 7-band images.

	k-means		Graph Cut		Mean Shift		SF	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.742	0.715	0.754	0.717	0.710	0.714	0.711	0.714
ARI	0.167	0.125	0.234	0.132	0.266	0.176	0.267	0.176
AMI	0.176	0.183	0.193	0.190	0.195	0.209	0.196	0.209

Table 6.4: Performance values for distance learning, Segmentation Weighting and Semi-supervised Segmentation Fusion algorithms on Training (Tr) and Test data.

	DL		QD		SW		SSSF	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.713	0.710	0.752	0.724	0.765	0.721	0.801	0.733
ARI	0.270	0.180	0.262	0.178	0.293	0.220	0.326	0.236
AMI	0.195	0.205	0.198	0.211	0.215	0.213	0.220	0.219

In the third set of experiments, k-means algorithm is employed on each band of 12-band Moderate Dimension Image: June 1966 aircraft scanner Flightline C1 (Portion of Southern Tippecanoe County, Indiana) [180]. The size of the image is 949×220, and there are 11 segments in the ground truth of the image [180]. The classes are background, Alfalfa, Br Soil, Corn, Oats, Red Cl, Rye, Soybeans, Water, Wheat, Wheat2. 104392 pixels are randomly selected for training and the remaining 104388 pixels are randomly selected for testing. In order to conserve the spatial distribution of the selected pixels, the pixels which reside in a segment with the same label in a spatial neighborhood are selected as test and training data. The distributions of pixels in training and test datasets are shown in Figure 6.7.a and Figure 6.7.b.



(a) Distribution of pixels in training dataset. (b) Distribution of pixels in test dataset.

Figure 6.7: Distribution of pixels given the segment labels in Moderate Dimension Image.

The results on the test data are given in Table 6.5, 6.6 and 6.7. $\hat{C} = 11$ and $\hat{\beta} = 0.9$ are obtained with $ARI = 0.004$ using the training data and with $ARI = 0.003$ for test data. It is observed

that the performance values for **SF** are smaller than the average performance values of base-layer segmentation outputs. Since the distance functions are computed for each segmentation pair, better performance for distance learning algorithms (**DL** and **QD**) is achieved. When prior information is employed using **SW** and **SSSF**, it is observed that the smaller weights are assigned to the segmentations with relatively small performance values. In addition, the output images of **SW** and **SSSF** are closer to the target segmentations obtained from the ground truth images. In summary, remarkable performance increases are observed in SSSF algorithm.

Table 6.5: Performance of k-means algorithms for Moderate Dimension Image.

Channel Number	**Training**			**Test**		
	RI	**ARI**	**AMI**	**RI**	**ARI**	**AMI**
1	0.533	0.008	0.154	0.537	0.014	0.153
2	0.533	0.009	0.134	0.531	0.006	0.134
3	0.532	0.003	0.124	0.528	0.009	0.126
4	0.529	0.010	0.142	0.532	0.009	0.140
5	0.529	0.005	0.131	0.532	0.006	0.131
6	0.531	0.000	0.110	0.523	-0.003	0.112
7	0.530	0.005	0.110	0.529	0.000	0.110
8	0.532	0.007	0.152	0.531	0.008	0.158
9	0.536	0.013	0.169	0.534	0.015	0.176
10	0.528	-0.002	0.123	0.527	-0.003	0.129
11	0.541	0.019	0.158	0.540	0.023	0.162
12	0.539	0.020	0.159	0.540	0.018	0.155

Table 6.6: Performance of the algorithms for Moderate Dimension Image.

	Average Base		**SF**		**DL**		**QD**	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.533	0.532	0.532	0.530	0.533	0.533	0.535	0.530
ARI	0.008	0.009	0.007	0.007	0.013	0.011	0.010	0.011
AMI	0.139	0.141	0.124	0.120	0.123	0.121	0.123	0.124

Table 6.7: Performance of the algorithms for Moderate Dimension Image.

	SW		**SSSF**	
	Tr	**Test**	**Tr**	**Test**
RI	0.545	0.540	0.553	0.550
ARI	0.020	0.023	0.109	0.110
AMI	0.168	0.176	0.177	0.185

6.3.2 Analyses on Aerial Images

In this section, the suggested algorithms are analyzed on aerial images collected from Google Maps. In the first set of experiments, the algorithms are employed on an image dataset consisting of four images which contain substations. The sizes of Image 1, Image 2 and Image 3 are 1060×922 and the size of Image 4 is 1076×922. In the ground truth images, white pixels represent the substations and black pixels represent the background (see Figure 6.8)

The algorithms are implemented four times; three images are selected as training images and the remaining image is selected as test image, at each time. In the results, the average performances are given. The results of the base-layer segmentation algorithms are given in Table 6.8. Table 6.9 shows the results of the proposed algorithms.

In the experiments, $\hat{C} = 2$ and $\hat{\beta} = 0.9$ are obtained with $ARI = 0.024$. The highest AMI and ARI performances are obtained from the outputs of Mean Shift and Graph Cut, respectively. In addition, ARI and RI values of the segmentations obtained from k-means algorithm is close to the performances of Graph Cut. Note that the performances obtained in this experiment is less than the performances obtained in the experiments which use two patched images of a single multi-spectral image given in the previous sections. This is observed because of the difference between the statistical properties of training and test images used in this section, such as the distributions of pixels and segments in the segmentations, which affects the parameter estimation performances of algorithms.

Although the performances of base-layer segmentation algorithms decrease in this experiment, the performances of the proposed algorithms increase according to the experiments on multi-spectral images. One of the reasons of this observation is the similarity of the images in the dataset. For instance, three of the images in the dataset (Image 1, 3 and 4 shown in Figure 6.8.a, Figure 6.8.e and Figure 6.8.f), have more similar statistical properties such as the size of the segments, than Image 2 shown in Figure 6.8.c.

Therefore, if two of the similar images are used with Image 2 in the training dataset and the remaining image is used as the test image, then the base-layer segmentations whose parameters are estimated using training datasets, may not correctly predict the labels of the test image.

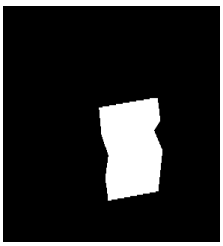
However, the proposed distance learning algorithms enable us learning the statistical properties of the segmentations and employ pair-wise distance computations between the segmentations to achieve a consensus among the segmentation outputs. In addition, **SW** and **SSSF** assign lower weights to the base-layer segmentations employed on Image 2. Therefore, the proposed algorithms provide higher performances.

Table 6.8: Performances of the popular image segmentation algorithms on Substation Image Dataset.

	k-means	Mean Shift	Ncut	Graph Cut
RI	0.579	0.219	0.545	0.584
ARI	0.137	0.018	0.022	0.148
AMI	0.109	0.216	0.085	0.123

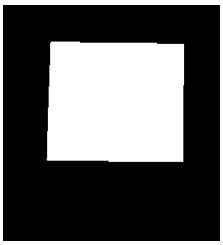
In the second set of experiments, the algorithms are employed on an image dataset, consisting of two images which contain airports, roads, urban area and green fields. The size of Image

(a) Image 1.



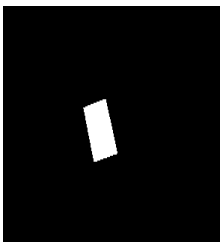
(b) GT of Image 1.

(c) Image 2.



(d) GT of Image 2.

(e) Image 3.



(f) GT of Image 3.

(g) Image 4.

(h) GT of Image 4.

Figure 6.8: Sample Images and Ground Truth (GT) Images in Substation Images Dataset .

Table 6.9: Performances of the suggested algorithms on Substation Image Dataset.

	SF	**DL**	**QD**	**SW**	**SSSF**
RI	0.520	0.535	0.585	0.605	0.602
ARI	0.024	0.144	0.140	0.153	0.167
AMI	0.016	0.078	0.110	0.230	0.246

1 is 1024×768 and the size of Image 2 is 1030×850. In ground truth images, white pixels represent airport, black pixels represent roads, green pixels represent green fields and red pixels represent urban area (see Figure 6.9). In this experiment, the algorithms are implemented two times and an images is selected as training images and the other image is selected as test image, at each time. In the results, the average performances are given.

As shown in Figure 6.9, the statistical properties of the images are similar. In the experiments, $\hat{C} = 3$ and $\hat{\beta} = 0.9$ are calculated with $ARI = 0.129$. The number of segments C is not correctly estimated because of the difference between the sizes of the segments, i.e. distributions of pixels in the segments. Since the segments labeled as urban area contain less pixels than the segments with the other segment labels, the pixels could not be correctly assigned to urban area segments. For instance, Mean Shift algorithm ignores the urban area segments by merging these segments to their adjacent segments. The results are given in Table 6.10 and 6.11.

In the results, we observe that the performances of the base-layer and the proposed segmentation algorithms employed on airport dataset are higher than that of the algorithms employed on substation dataset. One of the reasons of this observation can be explained as the statistical similarity of two images used in airport dataset. Moreover, considering the results of the experiments employed on multi-spectral and substation datasets, it can be stated that the performances of the proposed algorithms increase, as the statistical similarity between the images in training and test datasets increases.

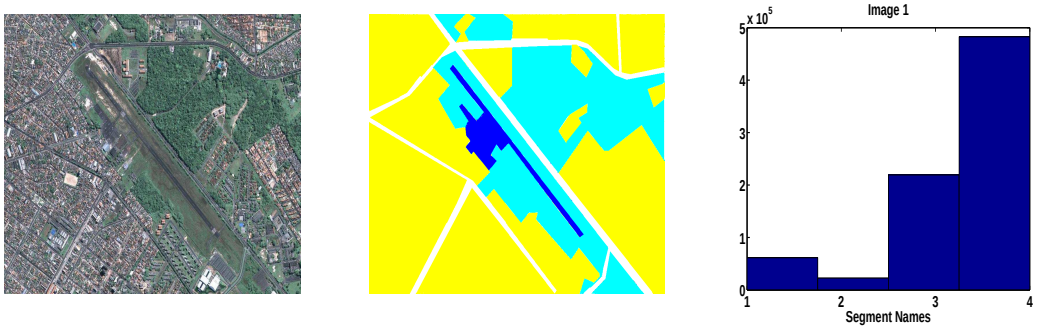
Table 6.10: Performances of the popular image segmentation algorithms on Airport Image Dataset.

	k-means	**Mean Shift**	**Ncut**	**Graph Cut**
RI	0.564	0.596	0.544	0.566
ARI	0.119	0.128	0.060	0.115
AMI	0.114	0.266	0.079	0.117

Table 6.11: Performances of the suggested algorithms on Airport Image Dataset.

	SF	**DL**	**QD**	**SW**	**SSSF**
RI	0.601	0.621	0.635	0.659	0.705
ARI	0.129	0.133	0.149	0.210	0.292
AMI	0.271	0.302	0.318	0.378	0.409

In the last set of experiments, the segmentation of roads in the aerial images is considered, which are analyzed in [178] and available on http://www.imageparsing.com/ Detailed infor-



(a) Image 1. (b) Ground Truth of Image 1. (c) Histogram of Image 1.

(d) Image 2. (e) Ground Truth of Image 2. (f) Histogram of Image 2.

Figure 6.9: Airport Images Dataset.

mation about the images in the dataset is given in [178, 179].

7 training and 7 test images with road and background labels are randomly selected from the dataset. The id numbers of the training and test images in the dataset are $tr = (7, 26, 40, 41, 42, 43, 77)$, and $te = (78, 90, 91, 92, 93, 94, 95)$, respectively. In order to observe the affect of the statistical similarity between training and test datasets, the performances are not averaged for different implementations of algorithms on random permutations of training and test images, and both of training and test performances are given in the results.

In the experiments, $\hat{C} = 3$ and $\hat{\beta} = 0.9$ are obtained with $ARI = 0.010$. The results are shown in Table 6.12 and Table 6.13. It is observed that the performances indices of **SF** are the same as the indices of Mean Shift. This is basically because of the fact that Mean Shift has a higher number of different segment labels than the other algorithms. Therefore, the outputs of Mean Shift suppress the outputs of other algorithms in the computation of distance functions. However, when **DL** and **QD** are employed, the distance functions are normalized, therefore, higher performances are obtained in Table 6.12. Moreover, higher performances than the base-layer segmentation algorithms are obtained in Table 6.13, when semi-supervision (**SSSF**) and weighting (**SW**) are employed in segmentation fusion.

6.4 Chapter Summary

In this chapter, two types of experimental analyses are given. First, the proposed segmentation fusion algorithms have been experimentally analyzed using synthetically generated datasets.

Table 6.12: Performances of k-means, Graph Cut, Mean Shift and Segmentation Fusion algorithms on Road Segmentation Dataset.

	k-means		**Graph Cut**		**Mean Shift**		**SF**	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.513	0.535	0.512	0.523	0.379	0.328	0.378	0.328
ARI	0.014	0.002	0.017	0.008	0.010	0.008	0.010	0.008
AMI	0.404	0.003	0.054	0.006	0.053	0.070	0.044	0.070

Table 6.13: Performances of distance learning, Segmentation Weighting and Semi-supervised Segmentation Fusion algorithms.

	DL		**QD**		**SW**		**SSSF**	
	Tr	**Test**	**Tr**	**Test**	**Tr**	**Test**	**Tr**	**Test**
RI	0.392	0.353	0.407	0.390	0.515	0.536	0.550	0.563
ARI	0.010	0.008	0.011	0.007	0.017	0.008	0.020	0.015
AMI	0.082	0.080	0.090	0.080	0.401	0.060	0.422	0.110

Then, performances of the proposed algorithms have been examined by fusing the outputs of four popular segmentation algorithms, namely, k-means, Mean Shift, Normalized Cuts and Graph Cut. The algorithms have been analyzed using multi-spectral and aerial images consisting of several objects, such as substations, airports and roads.

In the experiments on synthetic datasets, the relationships between the number of segments in the segmentations (C), the number of base-layer segmentations (K), RI and ARI performances of the proposed segmentation fusion algorithms are analyzed. In the generation of synthetic datasets, one of the essential assumptions of segmentation fusion which is that the elements of the segmentation set are the segments with noise added to a *target* consensus segmentation is used. The noise is defined as different labellings and segmentations of a randomly selected subset of pixels in a given base-layer segmentation s_i according to a noise level nl, which is the percentage of pixels replaced among the segments in s_i and a target segmentation s.

Experimental results show that RI and ARI values decrease as C increases when the difference between a target consensus segmentation s and base-layer segmentations s_i, $\forall i = 1, 2, \ldots, K$, is defined by a fixed noise level nl. In addition, RI and ARI values may have the same values as K increases if the number of segments in the segmentations is less than a fixed number, such as $k < 10$. However, the difference between RI and ARI values of different segmentations increases as k increases. In addition, the difference between RI and ARI values decreases as noise level nl increases.

For a fixed number of segments C, it is observed that the difference between RI and ARI values increases, as the number of segmentation outputs K increases. The experiments show that one of the reasons for the observation of this fluctuation is the early termination of the proposed SF and SSSF algorithms before a consensus segmentation is obtained.

In the experiments on real-world benchmark datasets, it is observed that the estimated $\hat{C}$ values are the same or similar to the C values that are defined by the users in the ground

truth data. However, $\hat{C}$ may not be a correct estimate of C of a training data if the statistical properties of the segments in the segmentation outputs are *different*. For instance, if there are 4 different segments with segment sizes 600000, 200000, 200000 and 1000 in the ground truth data, then the pixels in the last segment, which has the least number of pixels, may not be correctly placed in that segment. In other words, the labels of the pixels placed in that segment in the ground truth data may not be correctly predicted in the output segmentations of the segmentation fusion algorithms.

In addition, the experimental analyses show that the performances of the base-layer segmentation algorithms and the proposed segmentation fusion algorithms are sensitive to the statistical similarity of the images used in training and test datasets. The sensitivity of the base-layer segmentation algorithms affect the performances of the unsupervised Segmentation Fusion algorithm. However, distance learning methods can learn the statistical properties of the pixels and segments in the segmentation. Therefore, the proposed distance learning algorithms boost the performance of the unsupervised Segmentation Fusion (SF) algorithm. Moreover, the employment of semi-supervision on the proposed unsupervised SF and distance learning algorithms using Decision Weighting and Semi-supervised Segmentation Fusion algorithm further increase the performances.

In summary, the performances of the suggested algorithms demonstrate its efficacy in compromising over-segmented results and under-segmented results. Note that the performances of the proposed algorithms can be improved by the theoretical analyses on their open problems such as the investigation and modeling the dependency of the performances on the algorithm parameters, the statistical properties of the segmentations and images in training and test datasets. These theoretical problems are left to the future work, since the construction of performance bounds and mathematical modeling of the relationships between the algorithm parameters and data properties require novel mathematical approaches which enable us the integration of Statistical Learning Theory, Approximation Theory, Stochastic Optimization Theory and Theory of Computation.

CHAPTER 7

SUMMARY AND CONCLUSION

Over the last decade, Machine Learning methods are shifted from the single learning systems to ensemble learning architectures which are gathered under a decision fusion framework.

In this thesis, Decision Fusion approaches are studied from the optimization theoretic perspective. Based on the analysis of the optimization problems, which formalizes the decision fusion, a set of novel supervised, unsupervised and semi-supervised algorithms are introduced. The suggested decision fusion algorithms are assured to boost the performance of the individual learners which take place in the ensemble.

This thesis is organized in two parts: The first part involves the supervised decision fusion problem, whereas the second part is devoted to the unsupervised and semi-supervised image segmentation fusion problem.

In the first part, the classification error minimization problem has been reformulated for decision fusion applied to supervised learning. For this purpose, the classification error of a popular supervised learning algorithm, k-NN is decomposed into two components.

1. The first component of the error involves, the error difference between N-sample and large-sample error of k-NN classifier which is minimized using a distance learning method suggested by Short and Fukunaga [14] for a single classifier. Distance learning problem is reformulated as a decision fusion problem for the minimization of the error difference using a hierarchical decision fusion algorithm called Fuzzy Stacked Generalization (FSG).

2. The second component of the error involves the difference between large-sample error of k-NN and Bayes Error. Cover and Hart state that Bayes Error can be achieved if the samples are classified with complete *certainty* or *uncertainty*. In order to control the (un)certainty of the classification of a sample in base-layer and meta-layer classifiers, we first analyze the relationship between the classifiability of a sample by a base-layer classifier and a meta-layer classifier. For this purpose, a new distance function, called Decision Margin, is introduced. This function measures decision *certainty* of a base-layer classifier on the class membership of a sample and the contribution of this sample to meta-layer classification performance. The relationship between the performances of base-layer and meta-layer classifiers is analyzed by computing decision margins of samples. Since classification error of a meta-layer classifier and the expected value of decision margins of samples are linearly related, classification error of a meta-layer classifier can be minimized by minimizing decision margins of samples.

Based upon the analysis on the error differences mentioned above, first a decision fusion method using weighted decision margin minimization algorithm is proposed. Then, two sample selection algorithms are integrated to the FSG architecture. The suggested Supervised Decision Fusion architecture is tested and analyzed using synthetic and benchmark datasets and the results are compared with the results of state of the art algorithms such as Adaboost, Rotation Forest and Random Subspace.

The results show that the suggested FSG together with the weighted decision fusion and the sample selection methods, outperform the Adaboost algorithm in $C \geq 3$ class classification problems, and the Rotation Forest and Random Subspace algorithms for any C number of classes in all of the experiments. Specifically, weighted decision fusion and sample selection algorithms has a great impact for boosting the performance of the FSG, in the experiments.

Furthermore, the Supervised Decision Fusion method is applied to solve two real-world problems: First, FSG is applied on the remotely sensed image to detect the buildings. Second, it is applied to audio-visual data to detect the moving targets in an indoor environment. In these applications, it is observed that the supervised fusion method, FSG, boosts the performance of individual base-layer classifiers. In addition, it is observed that the class conditional entropies of the features are decreased through the layers of the FSG. In other words, fusing the decisions of the classifiers at the base-layer, extracts the discriminative information for sample classification from the features while transforming the feature spaces of the features extracted from observations to a feature space consisting of the decisions of base-layer classifiers.

There are three major superiorities of the suggested supervised decision fusion method: First of all, it minimizes the error difference between N-sample and large-sample error of k-NN classifiers. Second, the error difference between large-sample error of k-NN and Bayes Error can be minimized using weighted decision fusion and sample selection algorithms. Finally, base-layer classifiers can be trained as expert classifiers on each different feature space using the Decision Margin. Therefore, individual heterogeneous feature spaces which provide different information about the samples using the observations on data with different modalities can be fused efficiently for classification.

Yet, there are many challenges remain to be exploited for the suggested supervised decision fusion method: Among them, one of the most important issues is the selection of the feature spaces of the base-layer classifiers. Unfortunately, the performance of the suggested decision fusion method is very much dependent on the selection of the feature space of the base-layer classifiers. It may not be superior to the other state of the art algorithms for the classification of samples in a single feature space, if the feature spaces are not complimentary of each other. For instance, if two sets of features recognizes the same set of samples, but fail to recognize the remaining ones, the performance of the individual classifiers cannot be boosted by the FSG architecture. Another serious problem is the curse of dimensionality of the fusion space, which occur for large class classification problem when the training samples are not *sufficient*. For instance, if a feature space is 2 dimensional and the samples belong to $C = 10$ different classes, then the dimension of Fusion Space of the FSG is 20. Therefore, a curse of dimensionality problem occurs for the meta-classifier. FSG requires large number of training samples to provide high classification performance since fuzzy k-NN classifiers are used at the base-layer. Although FSG bridges the gap between N-sample and large-sample error of k-NN, higher large-sample errors are observed in the classification with less number of samples.

Sample selection algorithms are used to converge the large-sample error to Bayes Error. How-

ever, the algorithms may suffer from another problem, which is the observation of singularities in the feature spaces caused by the elimination of samples from the datasets. In other words, the singularity assumption of the sample selection algorithms may fail. If this assumption fails, then posterior probabilities cannot be estimated accurately by the base-layer classifiers and high classification errors are occurred at the meta-layer classifier. On the other hand, decision fusion using weighted decision margin minimization algorithm converges to an *optimal* solution, i.e. provides an *optimal* weight vector, if the solution exists and can be achieved using the training dataset. If an *optimal* solution is achieved, then the classification error of FSG converges to Bayes Error. Finally, analyzing the convergence properties, such as convergence rates and error bounds, of weighted decision fusion and sample selection algorithms and the conditions on the feature spaces required to achieve high classification performances is another challenge.

In the second part of the thesis, unsupervised image segmentation problem has been analyzed for decision fusion. A specific realization of segmentation fusion, called consensus segmentation, is introduced. The aim of this realization is to achieve a consensus among different segmentation outputs obtained from different segmentation algorithms. In order to provide an approximate solution to this problem with less computational complexity (i.e. in linear time), an unsupervised decision fusion algorithm called Segmentation Fusion (SF) is introduced using a stochastic optimization method called Filtered Stochastic BOEM. The assumptions on the data statistics have been relaxed by incorporating prior information on the statistical properties of the data, such as pixel and segment distributions, to estimate the number of segment labels and algorithm parameters and solve distance learning problems.

Semi-supervision is incorporated in SF using a new algorithm called Semi-supervised Segmentation Fusion (SSSF). In SSSF, side information about the co-occurrence of pixels in the same or different segments is formulated as the constraints of a convex optimization problem. Although the employment of semi-supervision by integrating prior and side information in SF boosts the consensus performance, computational complexity increases to polynomial time in the number of different segment labels C and number of pixels N. Therefore, a trade-off between algorithm performance and running time occurs.

Before concluding the thesis, we would like to comment on the relationship between group level sparsity used in Chapter 3 for Supervised Decision Fusion and ℓ_1 norm sparsity used in Chapter 5 for Semi-supervised Decision Fusion. Since this relation is expected to guide the researchers who works in decision fusion area, we think it is worth to spend a short section on this topic.

7.1 Future Work for Appraising the Sparsity in Decision Fusion

The relationship between weight regularization using ℓ_2 norm and ℓ_1 norm sparsity, i.e. Lasso, has been analyzed by Yuan and Lin to induce Group-wise sparsity to the weights, i.e. Group Lasso [111].

Let us denote the weight vectors used in Decision Fusion for Supervised Learning problem with $\bar{w}_j^{su} \in \mathbb{R}^C$, $j = 1, 2, \ldots, J$ which are assigned to decisions of base-layer classification algorithms, e.g. class memberships, to solve a decision margin minimization problem. Similarly, in semi-supervised segmentation fusion problem, we denote a weight variable with $w_i^{us} \in \mathbb{R}$, $i = 1, 2, \ldots, K$ each of which is assigned to a base-layer segmentation decision.

In other words, vectors of weights $\bar{w}^{su}$ are assigned to decision vectors of base-layer classifiers and variables of weights w^{us} are assigned to the values of distance functions computed between base-layer segmentations and a computed segmentation, for Decision Fusion using Supervised Learning and Semi-supervised Learning, respectively.

Group Lasso algorithm use a penalty term $\sum_{j=1}^{J} \|\bar{w}_j^{su}\|_2$ in the definition of the error function of the supervised decision fusion problem (3.29). In Lasso, ℓ_1 norm $\| \bar{w}^{us} \|_1$ of $\bar{w}^{us} = [w_i^{us}]_{i=1}^{K}$ is used as the penalty term. If $w_j^{su} \in \mathbb{R}^1$ is used in (3.29), i.e. a weight variable is assigned to each classifier for its decision on each class, then Group Lasso problem becomes a Lasso problem.

Mathematically speaking, the weight variables w_i^{us} are linearly dependent on each other when ℓ_1 norm regularization is used. On the other hand, the variables w_{jc}^{su}, $\forall c = 1, 2, \ldots, C$ of a weight vector $\bar{w}_j^{su}$ assigned to the decision of a base-layer classifier are non-linearly related to each other.

However, there is a linear relationship between a weight vector $\bar{w}_1^{su}$ assigned to a base-layer classifier and the variables w_{2c}^{su}, $\forall c = 1, 2, \ldots, C$ of another weight vector $\bar{w}_2^{su}$ [111]. Then, the penalty of Group Lasso used in supervised decision fusion remains in between ℓ_1 norm penalty term of Lasso used in semi-supervised segmentation fusion and ℓ_2 norm penalty term of a Tikhonov regularization problem observed in a special solution of Lasso, which is mentioned in Section 3.6.

Therefore, Group Lasso is used to induce sparsity to a group of decisions and Lasso is used to induce sparsity to an individual decision of a base-layer supervised classification and semi-supervised segmentation algorithm, respectively.

Sparse data processing has been popularly used in the machine learning community [177, 111, 192, 193, 194, 195]. On the other hand, there are various open problems such as the selection of appropriate sparsity level in the construction of an optimization problem or the convergence of the proposed solutions to the problem [196, 197, 198, 199, 200]. Although these problems are fascinating and their solutions may improve the performances of the suggested algorithms, detailed analyses of these open problems are out of the scope of this thesis and are considered as future work.

7.2 Passion of Machine Learning and Data Fusion

Epistemologically, the challenges of Statistical Learning methods and in particular, Decision Fusion approaches, can be studied under two headings: *i*) technical challenges and *ii*) conceptual challenges. Both challenges can be observed in theoretical and practical works. Technical difficulties of solving theoretical problems are usually observed in the development of *better* bounds for algorithm performances or computational complexities. For this purpose, new inter-disciplinary methods, such as stochastic optimization methods and probabilistic graphical models inspired from Physics, are used to solve the problems of Statistical Learning.

In order to achieve theoretical performance or computational complexity bounds of the algorithms implemented in real-world applications, we need to satisfy the theoretical requirements of the algorithms such as collecting impractically large number of samples. In addition, the

real-world systems which implement the algorithms may require the implementation of the algorithms in reasonable running times. By the development of new technologies such as internet and high precision and multi-modal sensors, the requirements on the sample sizes can be satisfied. By the development of new high performance computing platforms such as Cloud and GPU, algorithms with polynomial time complexity can be implemented in reasonable running times.

However, the fundamental conceptual problems such as defining class or segment labels, similarity or difference measures, choosing learning paradigms (e.g. algorithmic or statistical learning), approaches for a specific paradigm (e.g. Bayesian or frequentist approaches of statistical learning) or methods (e.g. transductive, discriminative or generative methods) for a given problem is still a challenge.

In real-world applications, the conceptual problems are approximated according to the requirements of the systems which solve the practical problems. Therefore, application specific solutions are proposed, for instance, labeling of the building pixels by the domain experts, in remote sensing applications, or approaching to the problem from the Bayesian or frequentist point of view if data and observations (hypothesis and learning models) are *fixed* (repeatable) or *repeatable* (fixed), respectively in large scale data mining applications.

In theoretical aspect, new conceptual paradigms are needed. For instance, *learning* problem may be redefined by analyzing the relationship between the properties of data, hypotheses, and algorithms that generate hypotheses using data extending Algorithmic and Statistical Learning Theories with new theories. Moreover, the fundamental concepts such as class, distance, similarity, and difference should be revised. For instance, the relationships between data, hypothesis and algorithms can be used to define the distance between two samples according to their (dis)similarity with respect to a criterion, as defined in maximum margin learning algorithms.

Besides providing approximate solutions to these existing problems, new technical and conceptual problems emerged by the development of new technologies mentioned above, such as decomposition of problems and algorithms, defining the *knowledge* of samples and extracting information to achieve the knowledge by mining large scale datasets.

In the end, we spent an effort to have approximate solutions to our daily problems for the sake of understanding the nature or understand the nature for the sake of having approximate solutions to our daily problems.

REFERENCES

[1] S. S. Carey, *A beginner's guide to scientific method.* Wadsworth Publishing Company, 2011.

[2] D. Bohm, *Thought As a System.* Routledge, Chapman & Hall, Incorporated, 1994.

[3] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[4] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* New York, NY, USA: Wiley, 2001.

[5] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory." in *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., vol. 3176. Springer, 2003, pp. 169–207.

[6] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning.* Cambridge, MA: MIT Press, 2006.

[7] O. Bousquet, U. v. Luxburg, and G. Ratsch, *Advanced Lectures On Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tubingen, Germany, August 4-16, 2003, Revised Lectures (Lecture Notes in Computer Science).* Springer Verlag, 2004.

[8] B. Dasarathy, *Decision fusion.* IEEE Computer Society Press, 1994.

[9] D. L. Hall and S. A. H. McMullen, *Mathematical Techniques in Multisensor Data Fusion (Artech House Information Warfare Library).* Norwood, MA, USA: Artech House, Inc., 2004.

[10] M. A. Abidi and R. C. Gonzalez, *Data fusion in robotics and machine intelligence.* San Diego, CA, USA: Academic Press Professional, Inc., 1992.

[11] D. L. H. James Llinas, Martin E. Liggins, *Handbook of Multisensor Data Fusion: Theory and Practice*, 2nd ed., ser. Electrical Engineering & Applied Signal Processing Series. Taylor & Francis, 2008.

[12] E. Fix and J. Hodges, *Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties.* USAF School of Aviation Medicine, 1951.

[13] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan 1967.

[14] R. D. S. II and K. Fukunaga, "The optimal distance measure for nearest neighbor classification," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 622–626, 1981.

[15] C. Senaras, M. Ozay, and F. Yarman Vural, "Building detection with decision fusion," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. PP, no. 99, pp. 1–10, 2013.

[16] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 176–190, Jan 2008.

[17] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the 8th International Conference on Computer Vision*, vol. 2, July 2001, pp. 416–423.

[18] A. Goder and V. Filkov, "Consensus clustering algorithms: Comparison and refinement," in *Proceedings of the Workshop on Algorithm Engineering and Experiments, ALENEX 2008, San Francisco, CA, USA, January 19, 2008*, J. I. Munro and D. Wagner, Eds. Workshop on Algorithm Engineering and Experiments, ALENEX 2008, San Francisco, CA, USA, Jan. 19, 2008, 2008, pp. 109–117.

[19] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[20] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[21] H. Daumé, III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, no. 1, pp. 101–126, May 2006.

[22] J. Jiang, "A literature survey on domain adaptation of statistical classiers," 2008.

[23] D. Tuia, E. Pasolli, and W. Emery, "Using active learning to adapt remote sensing image classifiers," *Remote Sensing of Environment*, vol. 115, no. 9, pp. 2232–2242, Sep 2011.

[24] L. Bruzzone and M. Marconcini, "Toward the automatic updating of land-cover maps by a domain-adaptation svm classifier and a circular validation strategy," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1108–1122, Apr 2009.

[25] D. Tuia, E. Pasolli, and W. J. Emery, "Dataset shift adaptation with active queries," in *2011 Joint Urban Remote Sensing Event (JURSE,2011)*, Munich, Germany, Apr 2011, pp. 121–124.

[26] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[27] ——, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[28] L. G. Valiant, "A theory of the learnable," in *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, ser. STOC '84. New York, NY, USA: ACM, 1984, pp. 436–445.

[29] L. Wu, S. C. H. Hoi, R. Jin, J. Zhu, and N. Yu, "Learning bregman distance functions for semi-supervised clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 3, pp. 478–491, Mar 2012.

[30] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," Department of Computer Science and Engineering, Michigan State University, Tech. Rep., 2006. [Online]. Available: www.cs.cmu.edu/~liuy/frame_survey_v2.pdf

[31] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[32] S. R. Kulkarni and G. Harman, "Statistical learning theory: a tutorial," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 3, no. 6, pp. 543–556, 2011.

[33] J.-H. Xue and D. M. Titterington, "Comment on "on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes"," *Neural Processing Letters*, vol. 28, no. 3, pp. 169–187, Dec 2008.

[34] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification using a hybrid generative/discriminative approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 712–727, Apr 2008.

[35] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, 1972.

[36] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Sep 2006.

[37] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, May 2009.

[38] S. Dasgupta and Y. Freund, "Random projection trees for vector quantization," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3229–3242, Jul 2009.

[39] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *Proceedings of the 3rd International Workshop on Algorithms and Computation*, ser. WALCOM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 274–285.

[40] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract)," in *Proceedings of the tenth annual symposium on Computational geometry*, ser. SCG '94. New York, NY, USA: ACM, 1994, pp. 332–339.

[41] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, Mar 2003.

[42] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Machine Learning*, vol. 52, no. 1-2, pp. 91–118, Jul 2003.

[43] F. G. Cozman, I. Cohen, and M. C. Cirelo, "Semi-supervised learning of mixture models," in *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, T. Fawcett and N. Mishra, Eds. Washington DC, USA: AAAI Press, 2003, pp. 99–106.

[44] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[45] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[46] T. Stathaki, *Image Fusion: Algorithms and Applications*. Academic Press, 2008.

[47] P. K. Varshney, *Distributed Detection and Data Fusion*, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996.

[48] F. Rottensteiner, J. Trinder, S. Clode, and K. Kubik, "Building detection by fusion of airborne laser scanner data and multi-spectral images: Performance evaluation and sensitivity analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 62, no. 2, pp. 135–149, June 2007.

[49] A. Turlapaty, B. Gokaraju, Q. Du, N. Younan, and J. Aanstoos, "A hybrid approach for building extraction from spaceborne multi-angular optical imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 1, pp. 89–100, Feb 2012.

[50] S. Fabre, X. Briottet, and A. Appriou, "Impact of contextual information integration on pixel fusion," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 9, pp. 1997–2010, 2002.

[51] F. R. Al-Osaimi, M. Bennamoun, and A. Mian, "Spatially optimized data-level fusion of texture and shape for face recognition," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 859–872, Feb 2012.

[52] C. C. Chibelushi, F. Deravi, and J. S. Mason, "A review of speech-based bimodal recognition," *IEEE Transactions on Multimedia*, vol. 4, no. 1, pp. 23–37, Mar 2002.

[53] J. R. Schott, *Remote Sensing : The Image Chain Approach: The Image Chain Approach*. Oxford University Press, USA, 2007.

[54] G. Healey and D. Slater, "Models and methods for automated material identification in hyperspectral imagery acquired under unknown illumination and atmospheric conditions," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 6, pp. 2706–2717, 1999.

[55] H. B. Mitchell, *Multi-Sensor Data Fusion: An Introduction*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[56] A. J. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. Chapman & Hall/CRC, 2007.

[57] M. Kirby, *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[58] A. N. Gorban, B. Kgl, D. C. Wunsch, and A. Zinovyev, *Principal Manifolds for Data Visualization and Dimension Reduction*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[59] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Journal Neural Networks*, vol. 13, no. 4-5, pp. 411–430, May 2000.

[60] H. Liu and H. Motoda, *Computational Methods of Feature Selection*, ser. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2007.

[61] L. Li, B. Zhang, and Y. Wu, "Fusing spectral and texture information for collapsed buildings detection in airborne image," in *IEEE International Geoscience and Remote Sensing Symposium*, Munich, Germany, July 2012, pp. 186–189.

[62] R. Hansch and O. Hellwich, "Random forests for building detection in polarimetric sar data," in *IEEE International Geoscience and Remote Sensing Symposium*, Honolulu, HI, July 2010, pp. 460–463.

[63] B. Fernando, E. Fromont, D. Muselet, and M. Sebban, "Discriminative feature fusion for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2012, pp. 3434–3441.

[64] S. Wender and K. Dietmayer, "A feature level fusion approach for object classification," in *Intelligent Vehicles Symposium, 2007 IEEE*, 2007, pp. 1132–1137.

[65] V. Sharma and J. Davis, "Feature-level fusion for object segmentation using mutual information," in *Augmented Vision Perception in Infrared*, ser. Advances in Pattern Recognition, R. Hammoud, Ed. Springer London, 2009, pp. 295–320.

[66] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[67] X. Huang and L. Zhang, "Comparison of vector stacking, multi-svms fuzzy output, and multi-svms voting methods for multiscale vhr urban mapping," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 261–265, 2010.

[68] N. Milisavljevic and I. Bloch, "Possibilistic versus belief function fusion for antipersonnel mine detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1488–1498, 2008.

[69] M. Fauvel, J. Chanussot, and J. Benediktsson, "Decision fusion for the classification of urban remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2828–2838, Oct. 2006.

[70] M. Ozay and F. T. Y. Vural, "A new decision fusion technique for image classification," in *Proceedings of the 16th IEEE the International Conference on Image Processing, (ICIP 2009)*, Cairo, Egypt, Nov 2009, pp. 2189–2192.

[71] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[72] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, Aug 1998.

[73] H. Yang, B. Ma, and Q. Du, "Decision fusion for supervised and unsupervised hyperspectral image classification," in *IEEE International Geoscience and Remote Sensing Symposium*, vol. 4, 2009, pp. 948–951.

[74] B. Zenko, L. Todorovski, and S. Dzeroski, "A comparison of stacking with meta decision trees to bagging, boosting, and stacking with other methods," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, ser. ICDM '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 669–670.

[75] Y. Fu, L. Cao, G. Guo, and T. S. Huang, "Multiple feature fusion by subspace learning," in *Proceedings of the 2008 international conference on Content-based image and video retrieval*, ser. CIVR '08. New York, NY, USA: ACM, 2008, pp. 127–134.

[76] S. Dai, M. Yang, Y. Wu, and A. K. Katsaggelos, "Detector ensemble," in *IEEE Conference on Computer Vision and Pattern Recognition*. Minneapolis, Minnesota, USA: IEEE Computer Society, June 2007.

[77] D. Parikh and R. Polikar, "An ensemble-based incremental learning approach to data fusion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 437–450, 2007.

[78] S. Avidan, "Ensemble tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[79] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proceedings of the 21st international conference on Machine learning*, ser. ICML '04. New York, NY, USA: ACM, 2004.

[80] S. Yu, T. Falck, A. Daemen, L.-C. Tranchevent, J. Suykens, B. De Moor, and Y. Moreau, "L2-norm multiple kernel learning and its application to biomedical data fusion," *BMC Bioinformatics*, vol. 11, no. 1, 2010.

[81] X. Sun, K. Fu, H. Long, Y. Hu, L. Cai, and H. Wang, "Contextual models for automatic building extraction in high resolution remote sensing image using object-based boosting method," in *IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, Boston, MA, USA, July 2008, pp. 437–440.

[82] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," in *Proceedings of the 14th International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 322–330.

[83] B. Waske and J. A. Benediktsson, "Fusion of support vector machines for classification of multisensor data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12-1, pp. 3858–3866, 2007.

[84] D. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[85] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, no. 1, pp. 271–289, May 1999.

[86] N. Ueda, "Optimal linear combination of neural networks for improving classification performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 207–215, Feb 2000.

[87] M. U. Şen and H. Erdogan, "Linear classifier combination and selection using group sparse regularization and hinge loss," *Pattern Recognition Letters*, vol. 34, no. 3, pp. 265–274, 2013.

[88] N. Rooney, D. Patterson, and C. Nugent, "Non-strict heterogeneous stacking," *Pattern Recognition Letters*, vol. 28, no. 9, pp. 1050–1061, 2007.

[89] G. Sigletos, G. Paliouras, C. D. Spyropoulos, and M. Hatzopoulos, "Combining information extraction systems using voting and stacked generalization," *Journal of Machine Learning Research*, vol. 6, pp. 1751–1782, Dec 2005.

[90] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine Learning*, vol. 54, no. 3, pp. 255–273, Mar 2004.

[91] X. Tan, S. Chen, Z.-H. Zhou, and F. Zhang, "Recognizing partially occluded, expression variant faces from single training image per person with som and soft k-nn ensemble," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 875–886, Jul 2005.

[92] S.-B. Cho and J. H. Kim, "Multiple network fusion using fuzzy logic," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 497–501, Mar 1995.

[93] L. I. Kuncheva, ""fuzzy" versus "nonfuzzy" in combining classifiers designed by boosting," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 729–741, Dec 2003.

[94] M. Ozay, "Performance analysis of stacked generalization," Ph.D. dissertation, Middle East Technical University, 2008.

[95] M. Ozay and F. T. Yarman-Vural, "A new fuzzy stacked generalization technique and analysis of its performance," *CoRR*, vol. abs/1204.0171, 2012.

[96] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, Sep 1992.

[97] K. E. Graves and R. Nagarajah, "Uncertainty estimation using fuzzy measures for multiclass classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 128–140, Jan 2007.

[98] M. Germain, J.-M. Boucher, and G. B. Benie, "Multiband image fusion using an unsupervised contextual method," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 2002, pp. 3341–3344.

[99] F. Salzenstein and A. O. Boudraa, "Unsupervised multisensor data fusion approach," in *Proceedings of the 6th International Symposium on Signal Processing and its Applications*, vol. 1, 2001, pp. 152–155.

[100] W. Pieczynski, "Unsupervised dempster-shafer fusion of dependent sensors," in *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, 2000, pp. 247–251.

[101] M. Haindl and S. Mikeš, "Unsupervised texture segmentation using multiple segmenters strategy," in *Proceedings of the 7th international conference on Multiple classifier systems*, ser. MCS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 210–219.

[102] F. Melgani and Y. Bazi, "Markovian fusion approach to robust unsupervised change detection in remotely sensed imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 4, pp. 457–461, 2006.

[103] T. Li and C. H. Q. Ding, "Weighted consensus clustering," in *2008 SIAM International Conference on Data Mining (SDM 2008)*, Atlanta, Georgia, 2008, pp. 798–809.

[104] Y. Zhang, E. Zeng, T. Li, and G. Narasimhan, "Weighted consensus clustering for identifying functional modules in protein-protein interaction networks," in *2009 International Conference on Machine Learning and Applications*, ser. ICMLA '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 539–544.

[105] L. Franek, D. D. Abdala, S. Vega-Pons, and X. Jiang, "Image segmentation fusion using general ensemble clustering methods," in *Proceedings of the 10th Asian conference on Computer vision - Volume Part IV*, ser. ACCV'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 373–384.

[106] S. Vega-Pons, X. Jiang, and J. Ruiz-Shulcloper, "Segmentation ensemble via kernels," in *First Asian Conference on Pattern Recognition (ACPR)*, 2011, pp. 686–690.

[107] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Consensus clustering: The filtered stochastic best-one-element-move algorithm," in *Conference on Information Sciences and Systems, The John Hopkins University*, Baltimore, MD, Mar 2011, pp. 1–6.

[108] H. Ohlsson and L. Ljung, "A convex approach to subspace clustering," in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, ser. CDC-ECC11, Orlando, FL, USA, 2011.

[109] F. Wang, X. Wang, and T. Li, "Generalized cluster aggregation," in *Proceedings of the 21st international Joint conference on Artifical intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1279–1284.

[110] J. Keller, M. Gray, and J. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on System, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580 –585, 1985.

[111] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[112] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan 2011.

[113] M. Ozay and F. T. Vural, "On the performance of stacked generalization classifiers," in *Proceedings of the 5th international conference on Image Analysis and Recognition*, ser. ICIAR '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 445–454.

[114] M. Ozay and F. Yarman Vural, "Linear separability analysis for stacked generalization architecture," in *IEEE 17th Signal Processing and Communications Applications Conference (SIU), 2009*, 2009, pp. 1009–1012.

[115] M. Ozay and F. Vural, "Analysis of feature concatenation operation on vector spaces," in *IEEE 18th Signal Processing and Communications Applications Conference (SIU), 2010.*, 2010, pp. 1–4.

[116] ——, "A theoretical analysis of feature fusion in stacked generalization," in *IEEE 17th Signal Processing and Communications Applications Conference (SIU), 2009.*, 2009, pp. 548–551.

[117] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 607–616, Jun 1996.

[118] E. Marchiori, "Class conditional nearest neighbor for large margin instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 364–370, Feb 2010.

[119] Y. Li and L. Maguire, "Selecting critical patterns based on local geometrical and statistical information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1189–1201, Jun 2011.

[120] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, Mar 2012.

[121] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, Dec 2005.

[122] C. B. D. Newman and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html

[123] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 221–228.

[124] V. Garcia, E. Debreuve, F. Nielsen, and M. Barlaud, "k-nearest neighbor search: fast GPU-based implementations and application to high-dimensional feature matching," in *IEEE International Conference on Image Processing (ICIP)*, Hong Kong, China, September 2010.

[125] R. E. Schapire, "A brief introduction to boosting," in *Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2*, ser. IJCAI'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1401–1406.

[126] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, Oct 2006.

[127] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

[128] H. Eidenberger, "Statistical analysis of content-based mpeg-7 descriptors for image retrieval," *Multimedia Systems*, vol. 10, no. 2, pp. 84–97, 2004.

[129] P. Salembier and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*, B. Manjunath, Ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[130] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov 2004.

[131] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on riemannian manifolds," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[132] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[133] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, 1st ed. Haboken, NJ, USA: Wiley-Interscience, 2004.

[134] P. Saeedi and H. Zwick, "Automatic building detection in aerial and satellite images," in *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision*, vol. 1. Hanoi, Vietnam: IEEE, 2008, pp. 623–629.

[135] B. Sirmacek and C. Unsalan, "Urban-area and building detection using sift keypoints and graph theory," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1156–1167, Apr 2009.

[136] A. O. Ok, C. Senaras, and B. Yuksel, "Automated detection of arbitrarily shaped buildings in complex environments from monocular vhr optical satellite imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, no. 99, pp. 1–17, Aug 2012.

[137] Q. Wang and Z. Jiang, "A grammatical framework for building rooftop extraction," in *IEEE International Geoscience and Remote Sensing Symposium*, vol. 3. Beijing, China: IEEE, 2009, pp. 334–337.

[138] M. Turker and D. K. San, "Building detection from pan-sharpened ikonos imagery through support vector machines classification," in *Proceedings of ISPRS Technical Commission VIII Symposium*, Kyoto, Japan, 2010, pp. 841–846.

[139] J. Inglada, "Automatic recognition of man-made objects in high resolution optical remote sensing images by svm classification of geometric image features," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 62, no. 3, pp. 236–248, Aug. 2007.

[140] E. Baltsavias, "Object extraction and revision by image analysis using existing geodata and knowledge: current status and steps towards operational systems," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 58, no. 3-4, pp. 129–151, Jan. 2004.

[141] D. Koc San, "Approaches for automatic urban building extraction and updating from high resolution satellite imagery," Ph.D. dissertation, Middle East Technical University, 2009.

[142] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1982.

[143] G. Mountrakis, J. Im, and C. Ogole, "Support vector machines in remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247–259, 2011.

[144] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[145] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, Jan 1975.

[146] X. Huang and L. Zhang, "An adaptive mean-shift analysis approach for object extraction and classification from urban hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 12, pp. 4173–4185, Dec 2008.

[147] I. Lee, L. Cheng, and R. Li, "Optimal parameter determination for mean-shift segmentation-based shoreline extraction using lidar data, aerial ortophotos, and satellite imagery," in *American Society for Photogrammetry and Remote Sensing (ASPRS) 2010 Annual Conference*, San Diego, USA, Apr 2010.

[148] C. Tucker, "Red and photographic infrared linear combinations for monitoring vegetation," *Remote Sensing of Environment*, vol. 8, no. 2, pp. 127–150, May 1979.

[149] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1975.

[150] M. Teke, E. Başeski, A. O. Ok, B. Yüksel, and C. Şenaras, "Multi-spectral false color shadow detection," in *Proceedings of the 2011 ISPRS conference on Photogrammetric image analysis*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 109–119.

[151] P. Zhong and R. Wang, "A multiple conditional random fields ensemble model for urban area detection in remote sensing optical images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 3978 –3988, Dec 2007.

[152] N. Suematsu, Y. Ishida, A. Hayashi, and T. Kanbara, "Region-based image retrieval using wavelet transform," in *Proceedings of the 15th International Conference on Visual Interface*, Calgary, Canada, May 2002, pp. 9–16.

[153] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, Nov 1973.

[154] I. Steinwart and A. Christmann, *Support Vector Machines*, ser. Information Science and Statistics. New York, NY, USA: Springer, 2008.

[155] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[156] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[157] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, May 2007.

[158] S. Aksoy, I. Yalniz, and K. Tasdemir, "Automatic detection and segmentation of orchards using very high resolution imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 8, pp. 3117–3131, 2012.

[159] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, Apr 2011.

[160] O. Lartillot and P. Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *Proceedings of the 10th International Conference on Digital Audio Effects*, Bordeaux, France, Sep 2007, pp. 237–244.

[161] K. F. Wallis, "A note on the calculation of entropy from histograms," Department of Economics, University of Warwick, UK, Tech. Rep., 2006.

[162] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[163] H. Zhang, J. E. Fritts, and S. A. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 260–280, 2008.

[164] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[165] N. X. Vinh and J. Epps, "A novel approach for automatic number of clusters detection in microarray data based on consensus clustering," in *Proceedings of the 9th IEEE International Conference on Bioinformatics and Bioengineering*, ser. BIBE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 84–91.

[166] J.-M. Beaulieu and M. Goldberg, "Hierarchy in picture segmentation: A stepwise optimization approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 150–163, Feb 1989.

[167] J. C. Tilton, "Analysis of hierarchically related image segmentations," in *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, Greenbelt, MD, 2003*, 2003, pp. 60–69.

[168] Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Multiple spectral-spatial classification approach for hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4122–4132, 2010.

[169] Y. Yang and K. Chen, "Temporal data clustering via weighted clustering ensemble with different representations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 2, pp. 307 –320, Feb 2011.

[170] T. Li, C. Ding, and M. I. Jordan, "Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization," in *Proceedings of the 7th IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 577–582.

[171] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, LNAI 3176, O. Bousquet and U. von Luxburg, Eds. Berlin: Springer Verlag, 2004, pp. 146–168.

[172] J. C. Tilton, *RHSEG User's Manual*, via email request to James.C.Tilton@nasa.gov, NASA, 2010.

[173] L. Kuncheva and S. Hadjitodorov, "Using diversity in cluster ensembles," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, The Hague, Netherlands, Oct 2004, pp. 1214–1219.

[174] C. A. Sugar and G. M. James, "Finding the number of clusters in a dataset," *Journal of the American Statistical Association*, vol. 98, no. 463, pp. 750–763, 2003.

[175] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 1073–1080.

[176] P. Luo, H. Xiong, G. Zhan, J. Wu, and Z. Shi, "Information-theoretic distance measures for clustering validation: Generalization and normalization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1249 –1262, Sep 2009.

[177] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[178] B. Yao, X. Yang, and S.-C. Zhu, "Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks," in *Proceedings of the 6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. EMMCVPR'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 169–183.

[179] J. Porway, Q. Wang, and S. C. Zhu, "A hierarchical and contextual model for aerial image parsing," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 254–283, Jun 2010.

[180] L. Biehl and D. Landgrebe, "Multispec: a tool for multispectral–hyperspectral image data analysis," *Computers and Geosciences*, vol. 28, pp. 1153–1159, Dec 2002.

[181] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, Aug 2000.

[182] A. P. Eriksson, O. Barr, and K. Åström, "Image segmentation using minimal graph cuts," in *Proceedings of SSBA Symposium on Image Analysis*, F. Georgsson and N. Börlin, Eds., 2006, pp. 45–48.

[183] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient n-d image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, Nov 2006.

[184] S. Bagon, "Matlab wrapper for graph cut," Dec 2006. [Online]. Available: http://www.wisdom.weizmann.ac.il/~bagon

[185] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, Sep 2004.

[186] Y. Boykov, O. Veksler, and R. Zabih, "Efficient approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1222–1239, Nov 2001.

[187] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, Feb 2004.

[188] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010.

[189] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, pp. 921–940, Oct 1988.

[190] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.

[191] F. Chung, *Spectral graph theory*, ser. CBMS Regional Conference Series in Mathematics, No. 92. American Mathematical Society, 1997.

[192] L. Wang and X. Shen, "1-norm multiclass support vector machines," *Journal of the American Statistical Association*, vol. 102, no. 478, pp. 583–594, 2007.

[193] M. Ozay, I. Esnaola, F. Yarman Vural, S. Kulkarni, and H. Poor, "Distributed models for sparse attack construction and state vector estimation in the smart grid," in *Proceedings of the 3rd IEEE International Conference of Smart Grid Communications*, Taiwan, 2012, pp. 306–311.

[194] ——, "Smarter security in the smart grid," in *Proceedings of the 3rd IEEE International Conference of Smart Grid Communications*, Taiwan, 2012, pp. 312–317.

[195] ——, "Sparse attack construction and state estimation in the smart grid: Centralized and distributed models," *IEEE Journal on Selected Areas in Communications - Special series on Smart Grid Communications*, 2013.

[196] F. R. Bach, "Consistency of the group lasso and multiple kernel learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, Jun 2008.

[197] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *ArXiv e-prints*, Jan 2010.

[198] H. Liu and J. Zhang, "Estimation consistency of the group lasso and its applications," *Journal of Machine Learning Research - Proceedings Track*, vol. 5, pp. 376–383, 2009.

[199] L. Jacob, G. Obozinski, and J.-P. Vert, "Group lasso with overlap and graph lasso," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 433–440.

[200] J. Liu and J. Ye, "Fast overlapping group lasso," *CoRR*, vol. abs/1009.0306, 2010.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Özay, Mete
Nationality: Turkish (TC)
Date and Place of Birth: 27.10.1983, Eskişehir
Marital Status: Single

EDUCATION

Degree	Institution	Year
V.S.R.C.	Department of EE, Princeton University, NJ, USA	2011-2012
M.S.	Informatics Institute, METU, Ankara, Turkey	2006-2008
B.S.	Department of Physics, METU, Ankara, Turkey	2001-2006

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2011-2012	Department of EE, Princeton University, NJ, USA	Visiting Researcher
2006-2013	Informatics Institute, METU, Ankara, Turkey	Research Assistant

PEER-REVIEW AND PROGRAM CHAIR ACTIVITIES

- IEEE Transactions on Signal Processing,
- International Journal of Remote Sensing, Taylor Francis,
- Journal of Statistical Papers, Springer,
- Journal of Pattern Recognition, Elsevier,
- IEEE International Symposium on Information Theory (ISIT),
- IEEE International Symposium on Smart Grid Communications (TPC Member),
- IEEE International Conference on Power and Energy (TPC Member),
- International Symposium on Computer and Information Sciences.

PUBLICATIONS

1. Mete Ozay, Fatos T. Yarman-Vural, A New Fuzzy Stacked Generalization Technique and Analysis of its Performance, to be submitted to IEEE Trans. on Neural Networks, previously submitted to IEEE TPAMI. (a preliminary version is available at http://arxiv.org/abs/1204.0171).

2. Mete Ozay, Haipeng Zheng, Fatos T. Yarman-Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Fusion of Segmentation Algorithms using Consensus Clustering, submitted to International Journal of Remote Sensing.

3. Mete Ozay, Fatos T. Yarman-Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Semi-supervised Consensus Clustering, submitted to IEEE Transactions on Geoscience and Remote Sensing.

4. Mete Ozay, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Fusion of Image Segmentation Algorithms using Consensus Clustering, ICIP, 2013.

5. Caglar Senaras, Mete Ozay, Fatos T. Yarman Vural, Building Detection with Decision Fusion, in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, no.99, pp. 1-10, March 2013.

6. Mete Ozay, Ilke Oztekin, Uygar Oztekin and Fatos T. Yarman Vural, Mesh Learning for Classifying Cognitive Processes, submitted to Human Brain Mapping.

7. Omer Ekmekci, Mete Ozay, Ilke Oztekin, Uygar Oztekin, Fatos T. Yarman-Vural, Mesh Learning for Object Classification using fMRI Measurements, IEEE International Conference on Image Processing (ICIP), 2013.

8. Itir Onal, Mete Ozay, Orhan Firat, Ilke Oztekin, Fatos T. Yarman Vural, Analyzing the Information Distribution in the fMRI measurements by estimating the degree of locality, The 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2013.

9. Orhan Firat, Mete Ozay, Itir Onal, Ilke Oztekin, Fatos T. Yarman Vural, Representation of Cognitive Processes Using the Minimum Spanning Tree of Local Meshes, The 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2013.

10. Orhan Firat, Mete Ozay, Itir Onal, Ilke Oztekin, Fatos T. Yarman Vural, Functional Mesh Learning for Pattern Analysis of Cognitive Processes, IEEE International Conference Series Cognitive Informatics and Cognitive Computing, 2013.

11. Mete Ozay, Inaki Esnaola, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Machine Learning Methods for Attack Detection in the Smart Grid, submitted to IEEE Transactions on Smart Grid.

12. Mete Ozay, Inaki Esnaola, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Sparse Attack Construction and State Estimation in the Smart Grid: Centralized and Distributed Models, accepted in IEEE Journal on Selected Areas in Communications - Special series on Smart Grid Communications.

13. Orhan Firat, Mete Ozay, Itir Onal, Ilke Oztekin, Fatos T. Yarman Vural, Functional Mesh Learning for Pattern Analysis of Cognitive Processes, IEEE International Conference Series Cognitive Informatics and Cognitive Computing 2013.

14. Mete Ozay, Inaki Esnaola, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Smarter Security in the Smart Grid, IEEE SmartGridComm 2012.

15. Mete Ozay, Inaki Esnaola, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Distributed Sparse Attacks and State Vector Estimation in the Smart Grid, IEEE SmartGridComm 2012.

16. Caglar Senaras, Baris Yuksel, Mete Ozay, Fatos T. Yarman Vural, Automatic Building Detection with Feature Space Fusion using Ensemble Learning, IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2012.

17. Orhan Firat, Mete Ozay, Itir Onal, Batuhan Karagoz, Ilke Oztekin,Fatos T. Yarman Vural, A Mesh Learning Approach for Brain Data Modeling, IEEE Signal Processing, Communication and Applications, 2012.

18. Mete Ozay, Haipeng Zheng, Fatos T. Yarman-Vural, Sanjeev R. Kulkarni, H. Vincent Poor, Consensus Segmentation for Level Selection and Fusion, ARO-MURI 4th Annual Meeting, January 10, 2012, University of California, Irvine, USA, 2012.

19. Baris Yuksel, Caglar Senaras, Mete Ozay, Fatos T. Yarman Vural, Automatic Building Detection From Satellite Images Using Stacked Generalization Architecture, IEEE Signal Processing, Communication and Applications, 2011.

20. Baris Yuksel, Caglar Senaras, Mete Ozay, Fatos T. Yarman Vural, Automatic Building Detection with Feature Space Fusion using Ensemble Learning, ISCIS, 2011.

21. Mete Ozay, Ilke Oztekin, Uygar Oztekin and Fatos T. Yarman Vural, Modeling cognitive states using machine learning techniques, 4th INCF Congress of Neuroinformatics, Boston, USA, 4 Sep - 6 Sep, 2011.

22. Mete Ozay, Fatos T. Yarman Vural, An Information Criterion for Kernel Matrix Combination Problem, International Symposium on Computer and Information Sciences (ISCIS), The Royal Society, University of College, London, 2010.

23. Mete Ozay, Fatos T. Yarman Vural, Analysis of Feature Concatenation Operation on Vector Spaces, IEEE Signal Processing, Communication and Applications, 2010.

24. Mete Ozay, Fatos T. Yarman Vural, A New Decision Fusion Technique for Image Classification, IEEE International Conference on Image Processing (ICIP), 2009.

25. Mete Ozay, Okan Akalin, Fatos T. Yarman Vural, Ensemble Learning For Multi Sensor Data Fusion With Applications On Human Motion Detection And Analysis, International Symposium on Computer and Information Sciences (ISCIS), 2009.

26. Mete Ozay, Fatos T. Yarman Vural, A Theoretical Analysis of Feature Fusion in Stacked Generalization, IEEE Signal Processing, Communication and Applications, 2009.

27. Mete Ozay, Fatos T. Yarman Vural, Linear Separability Analysis for Stacked Generalization Architecture, IEEE Signal Processing, Communication and Applications, 2009.

28. Mete Ozay, Tugba Taskaya Temizel, METU Campus Grid Application, 1st National High Performance and Grid Computing Conference, 2009.

29. Mete Ozay, Fatos T. Yarman Vural, On the Performance of Stacked Generalization Classifiers, International Conference on Image Analysis and Recognition (ICIAR), Lecture Notes in Computer Science: 445-454, Springer, 2008.

30. Mete Ozay, Fatos T. Yarman Vural, Performance Analysis of Stacked Generalization Classifiers, IEEE Signal Processing, Communication and Applications, 2008.