8-10-2018

# Efficient Data Driven Multi Source Fusion

Muhammad Aminul Islam

# Efficient data driven multi source fusion

By

Muhammad Aminul Islam

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2018

Efficient data driven multi source fusion

By

Muhammad Aminul Islam

Approved:

---

John E. Ball
(Major Professor)

---

Derek T. Anderson
(Co-Major Professor)

---

Nicolas H. Younan
(Committee Member)

---

James E. Fowler
(Committee Member/
Graduate Coordinator)

---

Jason M. Keith
Dean
Bagley College of Engineering

Name: Muhammad Aminul Islam

Date of Degree: August 10, 2018

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professors: John E. Ball and Derek T. Anderson

Title of Study: Efficient data driven multi source fusion

Pages of Study: 275

Candidate for Degree of Doctor of Philosophy

Data/information fusion is an integral component of many existing and emerging applications; e.g., remote sensing, smart cars, Internet of Things (IoT), and Big Data, to name a few. While fusion aims to achieve better results than what any one individual input can provide, often the challenge is to determine the underlying mathematics for aggregation suitable for an application. In this dissertation, I focus on the following three aspects of aggregation: (i) efficient data-driven learning and optimization, (ii) extensions and new aggregation methods, and (iii) feature and decision level fusion for machine learning with applications to signal and image processing.

The *Choquet integral* (ChI), a powerful nonlinear aggregation operator, is a parametric way (with respect to the *fuzzy measure* (FM)) to generate a wealth of aggregation operators. The FM has $2^N$ variables and $N(2^N - 1)$ constraints for $N$ inputs. As a result, learning the ChI parameters from data quickly becomes impractical for most applications. Herein, I propose a scalable learning procedure (which is linear with respect to training sample

size) for the ChI that identifies and optimizes only data-supported variables. As such, the computational complexity of the learning algorithm is proportional to the complexity of the solver used. This method also includes an imputation framework to obtain scalar values for data-unsupported (aka missing) variables and a compression algorithm (lossy or losselss) of the learned variables. I also propose a *genetic algorithm* (GA) to optimize the ChI for non-convex, multi-modal, and/or analytical objective functions. This algorithm introduces two operators that automatically preserve the constraints; therefore there is no need to explicitly enforce the constraints as is required by traditional GA algorithms. In addition, this algorithm provides an efficient representation of the search space with the minimal set of vertices. Furthermore, I study different strategies for extending the fuzzy integral for missing data and I propose a GOAL programming framework to aggregate inputs from heterogeneous sources for the ChI learning. Last, my work in remote sensing involves visual clustering based band group selection and Lp-norm multiple kernel learning based feature level fusion in hyperspectral image processing to enhance pixel level classification.

Key words: data/information fusion, multiple kernel learning, Choquet integral, genetic algorithm, support vector machine, classification, clustering, remote sensing, missing data, band grouping, data-driven learning, goal programming, fuzzy measure, fuzzy integral, optimization

DEDICATION

To my parents, Abu Taher Miah and Aklima Khatun.

ACKNOWLEDGEMENTS

I am very fortunate to have Professor Derek T. Anderson as my adviser (who has been my major professor until Fall 2017, when he moved to the University of Missouri). He has been very inspiring to me in my research and has always been encouraging and supportive of me. I am most grateful to him for his guidance, advice, and support. I shall forever remain indebted for all I have learned from him. This work would not be possible without his support.

I thank Professor John E. Ball for his support and it has been a privilege for me to work with him. I am grateful to Professor Nicolas H. Younan for providing opportunities to carry out my research. I am thankful to Professor James E. Fowler for his valuable suggestions during my graduate study.

The work presented in Chapter IV was done in collaboration with Dr. Anthony J. Pinar whereas Chapter X was done in collaboration with Bryce Murray. The works presented in Chapters II and XI were done in collaboration with Dr. Derek T. Anderson. I am also fortunate to have the opportunity to collaborate with Professor Timothy C. Haven, Dr. Fred Petry, Dr. Paul Elmore, Dr. Denson Smith, Professor Grant Scott, Professor Christopher Wagner, Xiaoxiao Du, and R. Marcum.

This work was partially supported by the Naval Research Laboratorys Base Program, Program Element No. 0602435N. This work was funded in part by the U.S. Army TACOM

Life Cycle Command under Contract No. W56HZV-08-C-0236 and the Army Research Office grant W911NF-16-1-0017 and W911NF-16-1-0241. Superior, a high performance computing cluster at Michigan Technological University, was used in obtaining some of the results presented in Chapter IV.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

## 1.1 Motivation

Countless existing and emerging technologies require the intelligent fusion of data or information–referred to hereafter as just data unless there is a specific reason to differentiate–arising from different sources. The most prevalent sources to date include humans, sensors and algorithms/machines. For example, unmanned aerial vehicles (UAVs) are being deployed for tasks like earth observations, precision agriculture and security and defense, to name a few. These systems typically engage in remote sensing using a multitude of sensors; e.g., thermal, visual, hyperspectral, SAR, LiDAR and/or structure from motion (SfM), etc. Our modern smart car thrust is also built on the basis of sensors to identify the roadway and obstacles in a variety of dynamic and complex environments. In areas like the internet-of-things (IoT), BigData and cyber-security, spatial, spectral and temporal data is arising from experts (humans), sensors and machines (algorithms processing data from humans and/or sensors). These are just a few examples of "multi-source" systems. The point is, fusion is at the heart of a great variety of our current technological advancements. However, fusion is a rather vague concept. If you ask two experts in two fields, or often within the same field, you will not get the same explanation for what constitutes fusion. Fusion is a variety of tasks that covers the most basics of making appropriate data

associations (e.g., which pixel in sensor X relates to what pixel in sensory Y) to underlying semantics (e.g., what do "values" on one domain mean relative to the other and how can we condition them so they can be combined) to aggregation functions (e.g., what laws do we use to transform our multiple inputs into an appropriate output?).

Due to the widespread interest in data fusion, several attempts have been made to define it. However, all have ultimately fallen short as they are too general or overly specific. For example, the Joint Directors Laboratory (JDL), which was a pioneer in formalizing the fusion process for the sake of eliminating redundancies and sharing knowledge across different branches of the DoD, gave the following definition of fusion in 1987; "a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete the timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results [197]." Picking on the definition for a moment, what do the following really mean; data versus information, why is correlation a necessary component, what is a state and why is it critical to fusion, why are situations and threats a part of fusion, what is a result, etc. The JDL even recognized that their definition was too restrictive and they revised it in 1998 with "Data fusion is the process of combining data to refine state estimates and predictions [174]." In the years following the JDL, different definitions were put forth, e.g., omnibus model (or the process model) [25], JDL model [174], the intelligence cycle [165], the Boyd Control loop [34], the waterfall model [24, 132], the Dasarathy model

[52], the omnibus model. However, the challenge remains, fusion means many things to different people in different contexts and no adequate unifying definition has emerged to date.

Herein, I investigate a focused part of "fusion", aggregation. The aim of this thesis is to investigate mathematical and algorithmic questions related to the laws that govern the transformation of multiple (potentially heterogeneous) inputs into an output. In general, the idea of aggregation is to obtain a "better" result than if we only used the individual inputs. However, better is not a well defined concept. In some applications, better might mean taking a set of inputs and reducing them into a single result that can be more efficiently or effectively used for visualization. Better could also refer to obtaining more desirable properties such as higher information content or lower conflict. In areas like machine learning and pattern recognition, theories that power the majority of engineering applications mentioned above, better often refers to some desirable property like more robust and generalizable solutions (e.g., classifiers). Herein, my focus is fusion as it relates to the combining of human information and also sensor data in the context of machine learning and signal processing. Specifically, this thesis is focused on novel (H1) efficient and scalable aggregation functions, (H2) their robust data-driven optimization and (H3) hyperspectral signal processing applications.

## 1.2 Contributions

Numerous approaches exist for aggregation. In this thesis, I focus on capacity theory and Choquet integration (ChI). The primary reason is that the ChI is a well-grounded and

powerful parametric aggregation framework that emerged out of Calculus. Depending on the conditions of the capacity, the ChI is capable of producing numerous common and exotic functions, such as the well known family of linear combinations of order statistics (LCOS). Furthermore, the ChI is a powerful framework because it has two parts, the modeling of interactions between inputs (the capacity) and a combination law for producing an output given a set of inputs (the integral). Whereas the ChI is a well-studied branch of mathematics, there are numerous challenges that remain unsolved. For example: what is the ChI on heterogenous data (e.g., non real-valued domains); what is the ChI for uncertainty capacities; how do we specify and/or learn the capacity and therefore the ChI; how do we efficiently store and compute the ChI; can the ChI combine heterogeneous data (e.g., probabilities and possibilities) and what does the result mean; how can the ChI fuse multi-spatial/spectral/temporal sensor data; among numerous other challenges. In order to support the identified H1, H2 and H3 tasks, I have published the following work.

1. (Binary Fuzzy Measures and Choquet Integration for Multi-Source Fusion [9])(H1, H2) In Chapter II, I discuss the binary fuzzy measure (aka normal and monotone capacity) and binary ChI. Specifically, I discuss its different properties and simple and efficient storage representations.

*D. T. Anderson, **M. Islam**, R. King, N. H. Younan, J. Fairley, S. Howington, F. Petry, P. Elmore, and A. Zare, "Binary Fuzzy Measures and Choquet Integration for Multi-Source Fusion," ICMT, 2017*

2. (The Fuzzy Integral for Missing Data [110])(H1) In Chapter III, I discuss different strategies for extending and adapting the fuzzy measure and ChI for missing data problems.

***M. Islam**, D. T. Anderson, F. Petry, D. Smith, P. Elmore, "The Fuzzy Integral for Missing Data," FUZZ-IEEE, 2017*

3. (Visualization and Learning of the Choquet Integral With Limited Training Data [151])(H1,H2) In Chapter IV, I provide ways to visualize and learn the ChI in light of limited training data.

*A. Pinar, T. Havens, **M. Islam**, D. T. Anderson, "Visualization and Learning of the Choquet Integral With Limited Training Data," FUZZ-IEEE 2017*

4. (CLODD based band group selection [105]) (H3) Chapter V is focused on the visual clustering tendency analysis of hyperspectral signal data for contiguous and non-continuous band group selection.

***M. Islam**, D. T. Anderson, J. E. Ball, N. Younan, "CLODD based band group selection," IGARSS, 2016*

5. (Fusion of Diverse Features and Kernels Using Lp-Norm Based Multiple Kernel Learning in Hyperspectral Image Processing [106])(H1,H3) Chapter VI focuses on feature level fusion of different band group segmentation's for pixel-level object detection in hyperspectral image processing. Focus is on how higher lp-norm MKL can help fuse and demonstrate variety across the band groupings.

***M. Islam**, D. T. Anderson, J. E. Ball, N. Younan, "Fusion of diverse features and kernels using lp-norm multiple kernel learning in hyperspectral image processing," WHISPERS, 2016*

6. (Multi-Criteria Based Learning of the Choquet Integral using Goal Programming [108]) (H1,H2) In Chapter VII, I discuss a flexible GOAL programming framework that allows for constrained optimization of the ChI in light of multiple sets of varying priority data and high-level expert knowledge.

***M. Islam**, D. T. Anderson, T. Havens, "Multi-Criteria Based Learning of the Choquet Integral using Goal Programming," submitted to North American Fuzzy Information Processing Society, pp. 1-6, Aug, 2015*

7. (Data-Driven Compression and Efficient Learning of the Choquet Integral [111]) (H1,H2) In Chapter VIII, I discuss a way to identify data supported variables in fusion and therefore data unsupported variables. Data supported variables can be individually optimized and strategies are identified for controlling (imputing) values for those variables that have never been encountered). I also provide lossly and lossless variable compression techniques.

***M. Islam**, D. T. Anderson, A. Pinar, T. Havens, "Data-Driven Compression and Efficient Learning of the Choquet Integral," IEEE Transactions of Fuzzy Systems, accepted Sept, 2017*

8. (Efficient Binary Fuzzy Measure Representation and Choquet Integral Learning [107]) In Chapter IX(H1, H2), I discuss an additional and improved way to think about, represent, compute, learn and store the binary capacity and subsequent ChI.

***M. Islam**, D. T. Anderson, X. Du, C. Wagner, T. C. Havens, Efficient Binary Fuzzy Measure Representation and Choquet Integral Learning, 2018 17$^{th}$ International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, accepted.*

9. (Explainable AI for Understanding Decisions and Data-Driven Optimization of the Choquet Integral [144]) Chapter X (H1) introduces new data-centric explainable AI tools for introspection of the ChI.

*B. Murray, **M. Islam**, A. J. Pinar, T. C. Havens, D. T. Anderson and G. Scott, Explainable AI for Understanding Decisions and Data-Driven Optimization of the Choquet Integral, FUZZ-IEEE 2018, accepted.*

10. (Fuzzy Choquet Integration of Deep Convolutional Neural Networks for Remote Sensing [13]) Chapter XI (H1) presents algorithms for fusing heterogeneous deep learning architectures using the ChI.

*D. T. Anderson and G. Scott, **M. Islam**, B. Murray, R. Marcum, Fuzzy Choquet Integration of Deep Convolutional Neural Networks for Remote Sensing, Computational Intelligence in Pattern Recognition, Springer-Verlag, accepted.*

In addition to the above published work, I have the following paper under review.

11. (An Efficient Evolutionary Algorithm for Optimization of the Choquet Integral [109]) In Chapter XII (H1, H2), I propose a Genetic algorithm with monotonicity and boundary preserving operations that facilitates an efficient optimization of the highly constrained ChI problems, both convex and non-convex.

***M. Islam**, D. T. Anderson, F. Petry, P. Elmore, An Efficient Evolutionary Algorithm for Optimization of the Choquet Integral, International Journal of Intelligent Systems, under review.*

# CHAPTER II

# BINARY FUZZY MEASURES AND CHOQUET INTEGRATION FOR MULTI-SOURCE FUSION

## 2.1 Introduction

Engineering applications like Big Data, remote sensing, unmanned vehicles, robotics, signal and image processing, and machine learning, to name a few, all have something in common. They all typically require the intelligent combining (aka fusion) of multiple *sources*. Here, the term source is used to refer to the "generator" of data or information, e.g., sensors, humans and/or algorithms. In general, the idea of fusion is to obtain a "better" result than if we only used the individual inputs. However, better is not a well defined concept. In some applications, better might mean taking a set of inputs and reducing them into a single result that can be more efficiently or effectively used for visualization. Better could also refer to obtaining more desirable properties such as higher information content or lower conflict. In areas like machine learning and pattern recognition, theories that power the majority of engineering applications mentioned above, better often refers to some desirable property like more robust and generalizable solutions (e.g., classifiers). Regardless of the task at hand or the particular application, fusion is a core tool at the heart of numerous modern scientific thrusts.

Whereas the implications of this article are far reaching, we narrow its vision and scope, without loss of generality, to the particular problem of multi-sensor fusion. It is practically impossible nowadays to provide an unbiased and comprehensive review of multi-sensor fusion as there have been a massive number of publications on this topic ranging from correspondence to aggregation mathematics, various types of uncertainty, missing data, tracking, and numerous other factors. A recent review is [122]. Multi-sensor fusion is also challenging to summarize because when multiple sensors are operating in different portions of the electromagnetic spectrum important domain information is relevant related to the physics of the sensors, phenomena (e.g., objects), environments and environmental conditions that they are operating in. Herein, we focus on the underlying set of mathematics that enable fusion across this broad spectrum of challenges. In general, an aggregation operator $f$ is a mapping that can take different *forms* (branches of mathematics). Usually, its a mapping of data (e.g., sensor measurements) from $N$ inputs, $X = \{x_1, ..., x_N\}$, to a single result, i.e., $f(h(x_1), ..., h(x_i), ..., h(x_N)) \in \Re$, where $h(x_i)$ is the data provided by input $x_i$. An introductory aggregation function text and guide for practitioners is [28].

Herein, we focus on the *Choquet integral* (ChI) [176, 44, 76, 8], a well-known and demonstrated parametric function for data and information fusion. The ChI is a generator function that is parametrized by the so-called *fuzzy measure* (FM), a monotone and normal capacity. Once the FM has been determined, either by an expert or learned from data, the ChI turns into a specific aggregation operator, e.g., a *linear combination of order statistics* (LCOS) [181]. The ChI has been used for numerous applications, to name a few; humanitarian demining [155], computer vision [181], pattern recognition [77, 80, 137, 118, 62],

multi-criteria decision making [70, 127], control theory [187], and multiple kernel learning [152, 150, 155, 98, 97]. Different approaches exist to learn the FM from data, e.g., [12, 121, 120, 136, 11, 26, 152].

In this article, we have two primary motives. The first is an application driven one. In [55], we studied the task of learning a ChI fusion for binary decision making problems, e.g., target versus non-target classification, relative to uncertainty in the labeling of training data. Our approach was to extend the ChI via *multiple instance learning* (MIL). Results were demonstrated for the fusion of multiple classifiers from one or more sensors relative to *explosive hazard detection* (EHD) for humanitarian demining and the fusion of RX detectors for hyperspectral image processing. In particular, we observed that the data learned ChI solutions almost always preferred answers that had FM variables in (or approximately) $\{0, 1\}$ versus a more arbitrary expected number on the standard $[0, 1]$ FM interval. An advantage is that learning the $2^N$ FM variables on $\{0, 1\}$ is a drastically simpler problem than learning on $[0, 1]$. This leads us to our second motivation. It is not only difficult to learn on $[0, 1]^{2^N}$ versus $\{0, 1\}^{2^N}$, but it is also a naturally intractable problem. Meaning, even for relatively small $N$ it can be a game stopper. Therefore, we are interested in multi-source problems that utilize a $\{0, 1\}$ versus $[0, 1]$ variable range since it is efficient to store only the one-valued terms and the ChI can easily be computed via a look up table on the fly.

Herein, we put forth the following contributions. First, we investigate a *binary FM* (BFM). Second, we define and study a *BFM ChI* (BChI). Third, we prove that for a BFM, two well-known *fuzzy integrals* (FIs), the *Sugeno integral* (SI) and ChI, are equivalent and therefore selection of integral is irrelevant. Fourth, we explain the underlying aggregation

philosophy for a BChI, a procedure we call the "best pessimistic agreement". Last, we show that only a small subset of one-valued variables need be stored, which leads to an efficient to store, learn and simple to compute via a look up table version of the ChI.

The remainder of this article is organized as such. In Section 2.2 we summarize the FM and ChI, Section 2.3 is an alternative way to think about the ChI, Section 2.4 describes a BFM and Section 2.5 outlines the BChI. Next, a semantic description of the BChI is provided (Section 2.6), followed by BChI memory savings (Section 2.7) and a fast way to calculate the BChI (Section 2.8).

## 2.2 Choquet Integral (ChI)

The reader can refer to [8] for a recent survey and description of theory, applications and important extensions of the *fuzzy integral* (FI) [176]. Let $X = \{x_1, x_2, ..., x_N\}$ be a set of $N$ inputs, e.g., two radar systems and an infrared sensor. A FM is a monotonic function defined on the power set of $X$, $2^X$, as $\mu : 2^X \rightarrow \Re^+$ that satisfies the following two properties: (i) boundary condition, $\mu(\emptyset) = 0$ and $\mu(X) > 0$ and (ii) monotonicity property, if $A, B \subseteq X$ and $A \subseteq B$, then $\mu(A) \leq \mu(B)$. Often an additional constraint is imposed on the FM to limit the upper bound to 1, i.e., $\mu(X) = 1$. Figure (2.1) is an illustration of the FM. FM variables of different cardinality are displayed with different colors.

Let $h(x_i)$ be the data/information, e.g., sensor measurement, from input $i$. The discrete ChI (finite $X$) is

$$\int_C h \circ \mu = C_\mu(h) = \sum_{i=1}^{N} h(x_{\pi(i)}) \left[\mu(A_i) - \mu(A_{i-1})\right], \tag{2.1}$$

where $\pi$ is a permutation of $X$, such that $h(x_{\pi(1)}) \geq h(x_{\pi(2)}) \geq \ldots \geq h(x_{\pi(N)})$, $A_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$, and $\mu(A_0) = 0$. Based on selection of $\mu$, the ChI turns into a specific aggregation operator. For example, when $\mu(A) = 0, \forall A \in 2^X \setminus X$, the ChI is equivalent to the minimum operator. When $\mu(A) = 1, \forall A \in 2^X \setminus \emptyset$, we obtain the maximum operator. When $\mu(A) = \frac{|A|}{N}, \forall A$ we obtain the arithmetic mean. More generally, when $\mu(A) = \mu(B)$ $\forall A, B \in 2^X$ such that $|A| = |B|$, we recover the familiar class of LCOS, e.g., min, max, soft min and max, mean, median, trimmed statistics, etc.

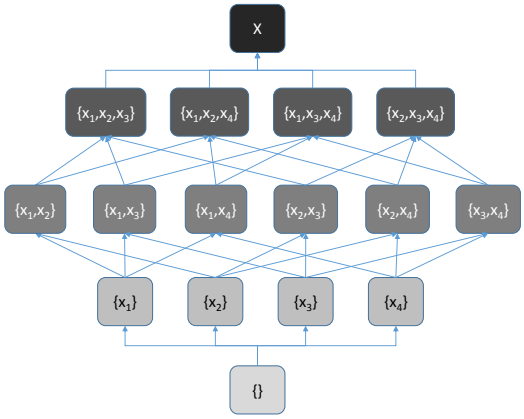



Figure 2.1
Illustration of variables and monotonicity constraints in the FM. Higher cardinality sets are shaded darker and arrows denote subsethood.

## 2.3 Lattice Representation

The first step in the discrete (finite $X$) ChI is to sort the inputs ($h$ values). In total, there are $N!$ possible sorts. An alternative way to think about the ChI is in terms of its underlying lattice, shown in Figure (2.1). The lattice is a way to relate the FM variables with respect to monotonicity, of which there are $N(2^{N-1} - 1)$ monotonicity constraints. An important

concept is a "walk" through the lattice. Figure (2.2) illustrates two different walks. Per a given input sorting, the ChI (Equation (2.1)) induces a walk, meaning it visits or uses $N + 1$ variables, i.e., $\mu(A_i) - \mu(A_{i-1})$. Formally, this is exactly one variable from each "level" in the FM, where level refers to all variables that have the same cardinality. This provides another convenient way to think about the ChI. The ChI is actually $N!$ different aggregation operators wrapped into one function—one for each walk through the lattice. In the case of operators like the minimum, maximum, average, and other LCOSs, each layer in the FM has constant measure value and therefore the operator is fixed and independent of which walk is observed. However, there are countless custom FMs (and therefore ChIs) that do not have this behavior, e.g., most cases of a BFM which we investigate later, and walks result in different aggregations per input value sort.

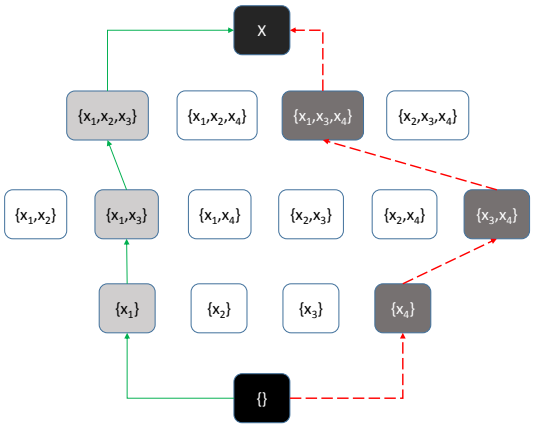



Figure 2.2
Example of two "walks" through the FM lattice with respect to the ChI. Variables visited in the first walk, i.e., for an input sorting of $h(x_1) \geq h(x_3) \geq h(x_2) \geq h(x_4)$, are shown in light gray and their order visited is indicated by green arrows. Variables visited in the second walk, i.e., $h(x_4) \geq h(x_3) \geq h(x_1) \geq h(x_2)$, are shown in dark gray and their order visited is indicated by dashed red lines. Note, the empty set and $X$ are both shaded in black and visited by both walks. Each walk is the set of weights used in the respective ChI calculation.

## 2.4 Binary Fuzzy Measure (BFM)

A BFM is trivial, it is a restriction of $\mu \in \{0, 1\}$ instead of $\mu \in [0, 1]$. While simple, a BFM has big implications. As already discussed, it simplifies the space for data-driven optimization, $\{0, 1\}^{2^N}$ versus $[0, 1]^{2^N}$, and it often leads to more compact solutions. When the FM is allowed to take any value in $[0, 1]$, we must generally store all variables. However, if we work with a BFM then naturally a number of these values, the exact number of depends on the target FM, are zero. Instead of storing all $2^N - 2$ variables (since $\mu(\emptyset) = 0$ and $\mu(X) = 1$), we can instead record just one-valued variables (which leads to savings in terms of storage). Figure (2.3) shows an example BFM.

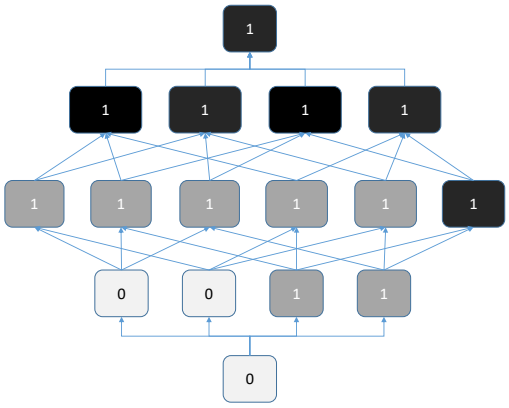



Figure 2.3
Example BFM—white cells have value zero, gray cells have value one (but one or more subsets, if we remove a single element, are zero) and black cells have value one (all direct subsets have value one).

## 2.5 BFM Choquet Integral (BChI)

Mathematically speaking, a *binary ChI* (BChI) is Equation (2.1). There is nothing new. However, what is interesting is what happens as a result of using a BFM. We can expand Equation (2.1) and observe the following (due to the monotonicity property of the FM).

$$\begin{aligned}\int_C h \circ \mu &= \sum_{i=1}^{N} h(x_{\pi(i)}) \left[\mu(A_i) - \mu(A_{i-1})\right], \\ &= \left(\sum_{i=1}^{n_1} h(x_{\pi(i)}) \left[0 - 0\right]\right) + h(x_{\pi(n_1+1)}) \left[1 - 0\right] \\ &+ \left(\sum_{i=n_1+2}^{N} h(x_{\pi(i)}) \left[1 - 1\right]\right),\end{aligned}$$

where $n_1$ is the number of inputs whose corresponding $\mu(A_i)$ terms are zero, $h(x_{\pi(n_1+1)})$ is the first term with $\mu(A_{n_1+1}) = 1$ and the remaining set are all one-valued as well. Note, without loss of generality, we partitioned the sum up as such, however we are more than aware that the first term may have an empty number of terms if $\mu(A_1) = 1$. What is important to note here is that due to monotonicity, a BFM will result in zero to $N - 1$ $(0 - 0)$ terms, a single $(1 - 0)$ weight and anywhere from zero to $N - 1$ $(1 - 1)$ weights. Thus, only one weight is one and all other weights are zero, resulting in the selection of a single input value, $h(x_{\pi(n_1+1)})$.

This leads us to Proposition 1. However, we first review the definition of the *Sugeno integral* (SI) [176],

$$\int_S h \circ \mu = S_\mu(h) = \bigvee_{i=1}^{N} \left(h(x_{\pi(i)}) \wedge \mu(A_i)\right), \tag{2.2}$$

where $\vee$ is the maximum operator and $\wedge$ is the minimum operator. However, these two operators, a t-conorm and t-norm can, and have, been extended beyond the scope of min-

imum and maximum. The SI and ChI are both FIs. We will not go into full depth here about similarities and differences between the two integrals. The reader can see [76] for more details. In general, the ChI is desirable because of factors such as; it does not use the minimum and maximum operators and is differentiable (a property that has been exploited before in data-driven learning), for an additive (probability) measure one recovers the classical Lebesgue integral, etc.

**Proposition 1** *The ChI and SI are equal for a BFM.*

Proof: This proof is trivial. As already noted, the BChI has a single non-zero weight and we get $h(x_{\pi(n_1+1)})$. When it comes to the SI, we take the minimum of each $h(x_{\pi(i)})$ with their $\mu(A_i)$. The SI can be expanded as such,

$$\begin{aligned}\int_S h \circ \mu &= \bigvee_{i=1}^{N} \big(h(x_{\pi(i)}) \wedge \mu(A_i)\big)\,, \\ &= \big((h(x_{\pi(1)}) \wedge 0) \vee (h(x_{\pi(2)}) \wedge 0) \vee ...\big) \\ &\vee (h(x_{\pi(n_1+1)}) \wedge 1) \\ &\vee \big((h(x_{\pi(n_1+2)}) \wedge 1) \vee ... \vee (h(x_{\pi(N)}) \wedge 1)\big)\,,\end{aligned}$$

where once again, without loss of generality, we denote a partitioning of the integral into parts, specifically two, where the first have zero-valued FM values and the second has one-valued FM values. However, if the first $A_1$ has value one then there are of course no zero terms. Note, $h(x_{\pi(n_1+1)})$ is the largest of the set $h(x_{\pi(n_1+1)})$ to $h(x_{\pi(N)})$. Therefore, it is selected as the result of the SI. This means that both the ChI and the SI both select the same input with respect to a BFM and are therefore equal, completing the proof. ■

What is interesting about Proposition 1 is it informs us that whereas there are numerous mathematical differences in general between the SI and ChI, many of which often impact a domain and dictate which integral is selected, they do not matter for the case of a BFM. The SI and the ChI are completely equal under this restriction.

### 2.6 What is the BChI Really Doing?

In this section, we explore what the BChI is doing in terms of underlying data/information aggregation philosophy. The BChI breaks down into the following steps.

1. Sort the input values from largest to smallest, i.e., $h(x_{\pi(1)}) \geq ... \geq h(x_{\pi(N)})$.
2. Find the smallest $i$ such that $\mu(A_i) = 1$.
3. Return $h(x_{\pi(i)})$.

Figure (2.4) is an illustration of the BChI. Note, the $h$ values are sorted in decreasing order and $g$ is monotonic. In addition, we can describe what the BChI is doing in a few (be it equivalent) ways.

**Interpretation 1:** The BChI is looking for the *first* $A_i$ (i.e., smallest $i$), relative to the $h$ sort, that has non-zero "worth" (FM value). The result is the "best pessimistic agreement" with respect to $h$ and $\mu$.

**Interpretation 2:** Another way to think about the BChI is the following. Each BFM variable that is equal to zero indicates a more-or-less *do not care* condition. Meaning, the input by itself should not be considered, more input (evidence) is needed before we are willing to make a decision. However, BFM variables equal to one indicate a group of

inputs is worth considering. Specifically, they all agree at least as much as their minimum value.

**Interpretation 3:** We can also talk about the BChI in terms of a lattice walk. The $N+1$ variables in a walk (the constants, i.e., non-$h$ terms, in the BChI) are a monotonic sequence of zeros followed by ones. Once we reach the first one value we take the minimum input for that set. Conversely, it is the maximum of all the inputs with corresponding BFM value one.

**Example 1:** As already stated, the focus of this article is the underlying mathematics of aggregation that are used by applications like multi-sensor fusion. For conceptual sake, consider a problem that requires interrogation in three different parts of the *electromagnetic* (EM) spectrum. For example, assume we have a task where $x_1$ is Radar, $x_2$ is *infrared* (IR) and $x_3$ is *EM induction* (EMI) for the handheld detection of buried explosive hazards in humanitarian demining [129, 196]. However, the following applies to any $N$ different spectral bands in a hyperspectral sensor or combination of single spectral band sensors in different parts of EM. Next, let there be one, for simplicity sake, respective algorithm on each of the above three sensors (GPR, IR and EMI) and let each sensor produce an output indicating target (value one) or non-target (value zero). The algorithms can be a binary decision, $\{0, 1\}$, or a $[0, 1]$ value generated by a probabilistic support vector machine, Bayes decision theoretic classifier, or any other number of countless pattern recognition or machine learning algorithms. Assume we already have our BFM—i.e., it was specified by an expert or learned from data (using any of the methods referenced in Section I). Let the BFM be $\mu(x_1)$=$\mu(x_2)$=$\mu(x_3)$=0 (no source is fully trustworthy by itself) and $\mu(\{x_2, x_3\})$=0. Let

all other variables be value one. Now, consider a particular set of inputs be $h(x_1) = 0.8$, $h(x_2) = 0.5$ and $h(x_3) = 0.01$. The BChI assigns a value of $0.5$ because when Radar is the highest value we require input and confirmation from IR (and disregard EMI). However, for an input of $h(x_3) = 1$, $h(x_2) = 0.9$ and $h(x_1) = 0.01$, the BChI gives us a value of $0.01$ because our BFM (relative to the ChI) informs us that when EMI has the strongest detection and IR is next, we need confirmation from all three sensors. The point is, the BFM, relative to the BChI, encodes a wealth of aggregation information across different walks about a particular task at hand.

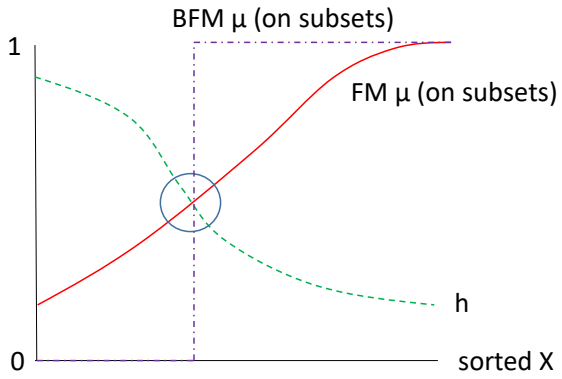



Figure 2.4
"Best pessimistic agreement" behavior of the BChI. Green dashed curve is the sorted $h$ values, red is the respective FM $\mu$ relative to sorted sets of increasing size and purple is a BFM $\mu$ relative to sorted sets of increasing size. The blue circle is the output of the BChI.

## 2.7 Compression of BFM for BChI

In Section 2.5, we discussed that for a BFM only one-valued terms need be stored. This can lead to big memory savings as there are otherwise $2^N - 2$ non-constant variables. For a relatively small problem with $10$ inputs we already have $1,022$ variables. For $20$ inputs,

we have $1,048,574$ variables. However, if a problem has a good number of one values in the lower layers of the lattice then savings diminish. This cannot be predicted, the exact amount of savings, in general. Each problem likely requires a different aggregation operator and therefore measure. Thus, the amount of savings depends on the problem at hand.

However, Figure (2.3) reveals a further savings. Variables shown in white (call it set $W \in 2^X$) are all zero-valued BFM terms. Values in gray (call it set $G \in 2^X$) are one-valued BFM terms that have at least input (all subsets if we remove one element) with value zero. Values in black (set $B \in 2^X$) are one-valued BFM terms whose entire input set are one-valued. It is trivial to show that $2^X = W \cup G \cup B$ and $W \cap G$ = $W \cap B$ = $G \cap B = \emptyset$. Furthermore, $G$ creates a partitioning between otherwise uninformative zero and one valued variables. What this means is, we have redundancy and given $G$ we can perfectly reconstruct the entire BFM. Furthermore, $G$ will often in practice be $|G| \ll |B \cup G \cup W|$. The take away is, one can identify and just store $G$ and perform the BChI.

## 2.8 BChI Data Structure

The ChI is typically calculated with respect to Equation (2.1), which has $N$ subtractions and $N$ multiplications and $N - 1$ additions. Alternatively, we can pre-compute all differences in $\mu$ variables (if storage is not an issue), which leads to only $N$ multiplications and $N - 1$ additions.

For the BChI, one option is to store the reduced set $G$ and on the fly, post sorting of our inputs, identify the corresponding term in $G$ and take the minimum of that set. There are a number of algorithms that can be employed. A computationally efficient, but possibly not the most memory efficient, scheme is to use a link list data structure. At most, we would take $N$ "steps" (traversals) in such a data structure (but likely far less on average). Figure (2.5) is an example for the BFM in Figure (2.3).

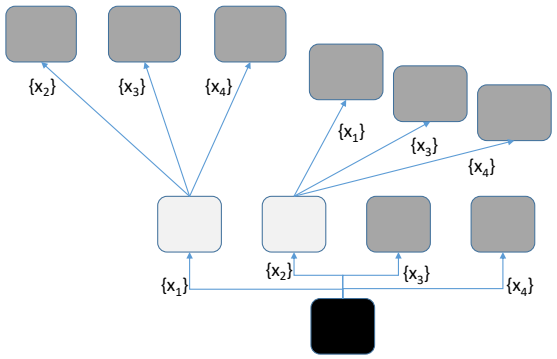



Figure 2.5
Example BChI link list data structure corresponding to Figure (2.3). The first step is to sort the inputs. Next, we start at the black node and move the direction of the largest input variable. If we ever encounter a "gray cell" (terminal node) then we take the minimum of the set of numbers up to that point. Otherwise, we keep following edges with respect to our sorting order until we hit a terminal node.

## 2.9 Conclusion and Future Work

In summary, the focus of this article is the underlying mathematics of data and information fusion. The tool selected is the *Choquet integral* (ChI), a flexible parametric aggregation function that morphs into a class of aggregation operators based on specification of the *fuzzy measure* (FM), a normal and monotone capacity. In practice, the FM is specified by an expert or learned by data. However, tractability (both in learning but also

memory storage) of the FM, and therefore the ChI, is of concern. In addition, applications like our discussed multiple instance learning ChI for binary target detection in multi-sensor systems (e.g., humanitarian demining and hyperspectral image processing) often learns or naturally prefers what we call a *binary FM* (BFM). Herein, we studied a *BFM ChI* (BChI). We discovered that a BFM renders the *Sugeno integral* (SI) and ChI equal, which is not often the case, and the underlying aggregation philosophy is the "best pessimistic agreement". We also observed that drastically fewer BFM variables need be stored and the BChI is nothing more than a simple look up strategy via some scheme like a link list data structure. Overall, we encounter both memory and computational savings. Last, learning a BFM is a much simpler task as the space is $\{0,1\}^{2^N}$ versus $[0,1]^{2^N}$.

Whereas the ChI is already a well-demonstrated theory for fusion, in future work we will explore the empirical benefits of using the proposed BChI for data-driven learning, ChI compression and efficient calculation in the context of specific applications for multi-sensor fusion.

# CHAPTER III

# THE FUZZY INTEGRAL FOR MISSING DATA

## 3.1 Introduction

Incomplete data and information–otherwise referred to hereafter as data unless there is a specific reason to differentiate–is intrinsic to some applications and external to others. For example, in geo-spatial systems, Big Data and the Internet of Things, to name a few, it is common that one or more sensors malfunction or become non-operational and yield no value or noisy data. In a medical context, all data for a diagnosis may not be available, especially expensive and invasive tests. In skeletal age-at-death estimation, remains may not be available due to natural or active intervention (unnatural death) causes. Similarly, companies often rely on incomplete consumer data to develop marketing strategies as consumers may not have enough knowledge–or decline to provide data–for parts of a survey. The point is, we are often faced with fusing and making intelligent robust decisions from incomplete data.

In rare cases missing data can be reacquired. However, often the *cost* of reacquiring outweighs/diminishes the net utility of that data. In [130], Little and Rubin created a taxonomy for missing data. They identified three classes; *missing at random* (MAR), *missing completely at random* (MCAR) and *missing not at random* (MNR). The complexity of missing data is further exacerbated by Donald Rumsfeld's quote–"There are known

knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we now know we don't know. But there are also unknown unknowns. There are things we do not know we don't know.[160]" The point is, missing data is not a trivial problem.

Before we dive into detail, it is important to review assumptions and *resources* that may or may not be available. These factors impact which procedure is selected. In some situations, no external knowledge is available and we are forced to use just what is observed. In other cases, historic data may be present. Other applications have access to other current observations where the present missing data is observed (similar to or the same as historic since data is available). Last, in some cases high-level a priori knowledge is available, e.g., a model (simulation or theoretical).

The bulk of missing data in *machine learning* (ML) literature is focused on topics like probabilistic graphical models (e.g., Bayesian nets), *support vector machines* (SVMs), regression or decision tree classifiers [159, 158, 141, 169, 65, 54, 156]. In [156], a decision tree classifier was extended by creating multiple test instances, where each took multiple paths along different splits of the missing attributes. A probability per leaf node was estimated based on the frequency of occurrence of training instances along the associated branch and the output was the class with the highest probability. In [161], Saar-Tsechansky and Provost learned multiple models for classification, where each model corresponds to an observed set of inputs encountered during training. Since the estimated model for a given set of observed input is optimal, it is expected to perform better than value based imputation but come at the cost of additional computation and storage. In soft-computing,

Zhong et al. constructed information granules around imputed data [214]–which can be interpreted as fuzzification of imputed scalars. Effectiveness was measured in two aspects, cover and specificity. Cover ensures that all the available information is utilized whereas specificity measures the extent of uncertainty, i.e., the range of the interval. In [214, 20], the *fuzzy-c means* (FCM) clustering algorithm was used for imputation. Data is partitioned using just observed data. The grade of membership of a missing sample is determined by substituting the membership of the nearest observed sample.

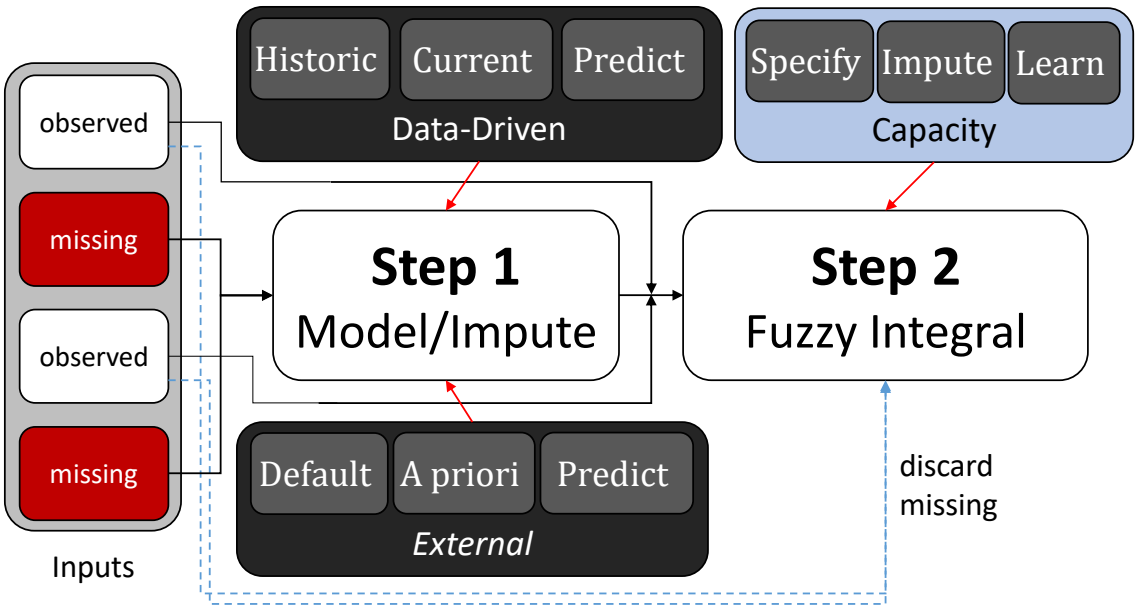



Figure 3.1
High-level illustration of main concepts (elaborated on in Section 3.3). The fuzzy integral uses observed data or observed and modeled/imputed data.

The *Choquet integral* (ChI), a type of *fuzzy integral* (FI), is a powerful parametric nonlinear aggregation function that has been used in numerous applications like machine learning, multi-sensor fusion and decision theory (e.g., multi-criteria decision making). To

the best of our knowledge, no attempt has been made to investigate missing data for the FI. Herein, we focus on two ways to extend the ChI to missing data (illustrated in Figure (3.1)). First, three normalizations are discussed relative to using just observed data. Second, a two step process, modeling/imputation and ChI extension, is discussed relative to computing with respect to missing data. In addition, an algorithm is put forth for learning the ChI relative to missing training data. Last, we explore the impact of these choices relative to different aggregation philosophies–selections of underlying *fuzzy measure* (FM).

## 3.2 ChI for Complete Data

In this section, necessary definitions are provided for the ChI and its data-driven learning relative to complete data. Let $X = \{x_1, \ldots, x_N\}$ be a set of finite elements, e.g., sensors, experts, criteria or attributes in decision making, or algorithms in pattern recognition. A FM is a monotonic set-valued function defined on the power set of $X$, $2^X$, as $\mu : 2^X \rightarrow \mathbb{R}^+$ that satisfies two properties; (i) boundary condition, $\mu(\emptyset) = 0$, and (ii) monotonicity, if $A, B \subseteq X$ and $A \subseteq B$, $\mu(A) \leq \mu(B)$. Often an additional constraint is imposed to limit the upper bound, i.e., $\mu(X) = 1$. Herein, without loss of generality we consider this condition for simplicity and convenience, which is useful in contexts like decision-level fusion.

Consider a training set with $M$ observations and labels, $O = \{(\mathbf{o}_j, y_j)\}, j = 1, 2 \ldots, M$, where $\mathbf{o}_j \in \mathbb{R}^N$ is the $j$th observation, $y_j \in \mathbb{R}$ is the associated label, and $o_j(x_k)$ corresponds to the observed value for $j$th instance and $k$th input. Let $\mathbf{u} = [\mu(\{x_1\}), \mu(\{x_2\}),$

$\ldots, \mu(X)]^T$ be the $2^N - 1$ dimensional vector containing all FM variables except $\mu(\emptyset)$. The discrete ChI on $\mathbf{o}_j$ with respect to $\mu$ is

$$C_\mu(\mathbf{o}_j) = \sum_{i=1}^{N} [o_j(x_{\pi_j(i)}) - o_j(x_{\pi_j(i-1)})]\mu(S_{\pi_j(i)}), \tag{3.1}$$

where $\pi_j$ is a permutation function for observation $\mathbf{o}_j$ on the indices, $\{1, 2, \ldots, N\}$, that satisfies $0 \leq o_j(x_{\pi(1)}) \leq \ldots \leq o_j(x_{\pi(N)})$, where $S_{\pi_j(i)} = \left\{x_{\pi_j(i)}, x_{\pi_j(i+1)}, \ldots, x_{\pi_j(N)}\right\}$ and $o_j(x_{\pi_j(0)}) = 0$ [30]. Equation (3.1) can be written as

$$C_\mu(\mathbf{o}_j) = \mathbf{c}_j^T \mathbf{u}, \tag{3.2}$$

where $\mathbf{c}_j$ is a column vector containing the $(2^N - 1)$ coefficients for observation $\mathbf{o}_j$. Let $k$ be the index of variable $\mu(B \in 2^X)$ in $\mathbf{u}$. Then the $k$-th element of $\mathbf{c}_j$ is $c_{jk} = o_j(x_{\pi_j(l)}) - o_j(x_{\pi_j(l-1)})$ if $\exists S_{\pi_j}(l) = B$ for $l \in \{1, \ldots, N\}$, and 0 otherwise. The monotonicity constraints can be written as $\mu(A) \leq \mu(A \cup q), \forall A \subset X$ and $\forall q \in X, q \notin A$. The *sum of squared error* (SSE) between the ChI for all the observations in the training data, $O$, and corresponding labels is

$$\begin{aligned} E_1(O, \mathbf{u}) &= \sum_{j=1}^{M} (C_\mu(\mathbf{o}_j) - y_j)^2 = \sum_{j=1}^{M} (\mathbf{c}_j^T \mathbf{u} - y_j)^2 \\ &= \sum_{j=1}^{M} (\mathbf{u}^T \mathbf{c}_j \mathbf{c}_j^T \mathbf{u} - 2y_j \mathbf{c}_j^T \mathbf{u} + y_j^2). \end{aligned} \tag{3.3}$$

Based on this, the least square minimization problem can be expressed as the *quadratic programming* (QP)

$$\text{(OP1)} \ \min_{\mathbf{u}} f_O(\mathbf{u}) = \mathbf{u}^T H \mathbf{u} + \mathbf{d}^T \mathbf{u},$$

$$\mu(A) \leq \mu(A \cup q), \quad \forall A \subset X \text{ and } \forall q \in X, q \notin A,$$

(monotonicity conditions) (3.4a)

$$\mu(\emptyset) = 0, \qquad \text{(boundary conditions)} \tag{3.4b}$$

$$\mu(X) = 1, \qquad \text{(normality conditions)} \tag{3.4c}$$

where $H = \sum_{j=1}^{M} \mathbf{c}_j \mathbf{c}_j^T$ and $\mathbf{d} = -2 \sum_{j=1}^{M} y_j \mathbf{c}_j$. This can be minimized by standard QP solvers. Herein, OP1 is used for the training while Equation (3.2) is used for prediction.

### 3.3 ChI for Missing Data

First, we establish some notation for the following subsections. For instance $\mathbf{o}_j$, inputs (data) are available for the set $X_j^w \subseteq X$ such that $x_i \in X_j^w, i \in I_j^w, I_j^w \subseteq \{1, 2, \ldots, N\}$ and data is not available for $X_j^h = X \setminus X_j^w$. Last, let the cardinality of $X_j^w$ and $X_j^h$ be $N_j^w$ and $N_j^h$ respectively.

**Remark 1.** We start by exploring if it is acceptable to calculate just the subset of Equation (3.1) associated with observed data. It turns out that doing so is equivalent to imputing zeros for the missing data. While it does not *mathematically break* the ChI, relative to imputed data, these zeros (or other constants at that) semantically impact aggregation and properties like idempotency (namely our expectations on it). There are better philosophies (following sections) that do not force us to inject potentially misleading/uninformed data and allows us to model and compute with respect to our uncertainty.

### 3.3.1 Method 1: ChI for Observed Data Only

In this subsection we restrict the ChI to only using available data and therefore the subset of the FM associated with our observed data. However, since the FM is defined on $N$, adding or removing inputs can (it is trivial to show) numerically and semantically alter the ChI and result in a measure that violates the properties of a FM, namely the upper boundary condition. We discuss three approaches to overcome this.

First, we define a "sub-ChI/FM" on instance $\mathbf{o}_j$, $C_{\mu_j^w}(\mathbf{o}_j)$, with respect to all available data ($X_j^w$),

$$C_{\mu_j^w}(\mathbf{o}_j) = \sum_{i=1}^{N_j^w} [o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})]\mu_j^w(S_{\pi_j^w(i)}), \tag{3.5}$$

where $\pi_j^w$ is the permutation operation for $\mathbf{o}_j$ performed on observed sources. A first idea is to let $\mu_j^w(A) = \mu(A), \forall A \subseteq X_j^w$. However, $\mu(X_j^w)$ may not be equal to 1. Thus, we have a boundary condition difference of

$$\Delta\mu_j^w(X_j^w) = 1 - \mu(X_j^w). \tag{3.6}$$

Whereas monotonicity is obviously preserved, important properties like boundedness and idempotency are not guaranteed for $C_{\mu_j^w}(\mathbf{o}_j)$. The problem resides in directly taking FM values from $\mu$. In order to realize a *legitimate* (property preserving) $C_{\mu_j^w}(\mathbf{o}_j)$ we need to calculate $\mu_j^w$ as a monotonic and boundary condition preserving function of $\mu$. By definition, $\mu(S_{\pi_j^w(i-1)}) \geq \mu(S_{\pi_j^w(i)})$, which gives the condition $\Delta\mu_j^w(S_{\pi_j^w(i-1)}) \geq \Delta\mu_j^w(S_{\pi_j^w(i)})$ on relative change in the FM. This condition along with Equation (3.6) sets the boundary for $\Delta\mu_j^w(S_{\pi_j^w(i-1)})$ as $0 \leq \Delta\mu_j^w(S_{\pi_j^w(i-1)}) \leq 1 - \mu(X_j^w)$. The question is, how do we determine the $\Delta\mu_j^w$ values?

*Upper boundary fix:* The first and simplest way to remedy the problem is to fix what is broken; $\mu_j^w(X_j^w) = 1$ and all other variables are $\mu_j^w(A) = \mu(A)$. This method satisfies the boundary conditions and monotonicity. However, an unwanted side effect, when $\Delta\mu_j^w(X_j^w) > 0$, is the introduction of a larger than previously modeled difference between the normalized $\mu_j^w(X_j^w)$ (now inflated) and $\mu_j^w(X_j^w \setminus x_k)$ ($\forall x_k \in X_j^w$) terms.

*Additive normalization:* Another idea is to add $\Delta\mu_j^w(X_j^w)$ to each FM variable, except for $\mu_j^w(\emptyset)$. This ensures boundary conditions and monotonicity. This procedure preserves the relative differences between consecutive (increasing cardinality) normalized $\mu_j^w$ variables, except for the difference between the $\mu_j^w(\{x_k\})$ ($\forall x_k \in X$) and $\mu_j^w(\emptyset)$. Furthermore, if $\mu(A) = 0$, $\forall A \subset X \setminus \emptyset$, then $\mu_j^w(A) \geq 0$ when $\Delta\mu_j^w(X_j^w) \geq 0$. That is, a subset that was previously given no weight/importance/utility is no longer zero valued.

*Uniform normalization:* The last fix considered is to divide each term by $\mu_j^w(X_j^w)$ (when $\mu_j^w(X_j^w) \neq 0$). This ensures boundary conditions, is monotonic and does not have the inflation side effects of upper boundary fix nor additive normalization. This scaling is semantically more justifiable.

**Remark 2. (Limitations of the observation-valued ChI)** An advantage of using just observed data is we are computing with respect to what we know and the result is bounded between the minimum and maximum observations, i.e., we will not infer values higher or lower that we were informed. However, this advantage is also its shortcoming. If the unobserved data is critical to the task at hand then it will obviously fall short as no attempt was made to model/impute the missing data and our associated uncertainty.

### 3.3.2 Method 2: Modeling and Imputing Missing Data

In this subsection, we highlight various approaches for modeling and imputing missing data. This is not a task unique to our paper nor is it a primary contribution of our work. This is actually a strong example of how the field of fuzzy set theory can or already does contribute to missing data problems. In general, we have the following categories.

1. **Discard**; inputs for missing data are ignored.
2. **Default**; a user/system/etc. defined value is used.
3. **Historic data**; prior complete or missing data is used.
4. **Current data**; data from other present instances where the missing data inputs are observed is used.
5. **A priori**; use high-level knowledge external to the data.
6. **Prediction**; use any of the above in combination with the current observed instance data to predict missing data.

Options three to six are research topics. Obviously, discarding and default are trivial. The task of prediction is beyond the scope of this work (see [63] for a recent review). Furthermore, a priori knowledge is task dependent and not explored herein. We touch on historic and/or current data (categories 3 and 4).

**Example 1. (Fitting a distribution to historic and/or current data)** When historic and/or other current instances are available we can fit a distribution to that data. However, what

distribution do we use? There are a number of distributions (e.g., normal, trapezoidal, etc.) and distribution fitting methods (e.g., construction of a type-2 fuzzy set [194], Gaussian mixture model, etc.). In general, the selection of theory depends on the *nature* of the underlying uncertainty. For example, often we follow the central limit theorem and model data via a normal distribution, which is fully determined by its mean and variance. The point is, there are numerous ways (beyond the ChI scope of this work) to derive distributions from historic and/or current observations.

**Example 2. (Interval modeling)** Another option is to model missing data as an interval, i.e., $\bar{o}_j(x_k) = [o_j^-(x_k), o_j^+(x_k)]$, $o_j^-(x_k) \leq o_j^+(x_k)$. However, as before, where does this interval come from? For observed data there is no uncertainty, i.e., $o_j^-(x_k) = o_j^+(x_k)$. For missing data, there are a number of possibilities. If uncertainty is modeled as a distribution then we can calculate $\beta\sigma_{x_k}$, i.e., $\bar{o}_j(x_k) = [m_{x_k} - \beta\sigma_{x_k}, m_{x_k} + \beta\sigma_{x_k}]$, where $m_{x_k}$ is the mean and $\sigma_{x_k}$ is the variance of $x_k$, and $\beta$ is an arbitrary constant. Alternatively, if nothing is known about the missing data and no assumptions are applicable, then minimum and maximum values can be used. In the extreme case, we can associate each missing input with $\bar{o}_j(x_k) = [0, 1]$, aka total ignorance.

### 3.3.3 Method 2: Using Modeled or Imputed Data

In subsection 3.3.2 we touched on different ways to model uncertainty. Note, we do not care about the underlying *meaning* (e.g., probability, possibility, etc.). In this subsection we tackle how to use such interval or set-valued data in the ChI. We start with the $\Re$-valued ChI, followed by the interval-valued ChI then type-1 and type-2 integrand-valued ChIs.

Figure (3.2) shows different options (pathways) between modeling/imputation choices and FI extensions. In general, there is no "winner". Ideally, in the theme of David Marr's Principle of Least Commitment and Principle of Graceful Degradation, we advocate the modeling of computation with respect to *full* (set-valued) data. However, some applications may impose restrictions (eliminate options). Furthermore, different applications might require type reduction if lower order uncertainty is desired or scalar-valued outputs are needed.

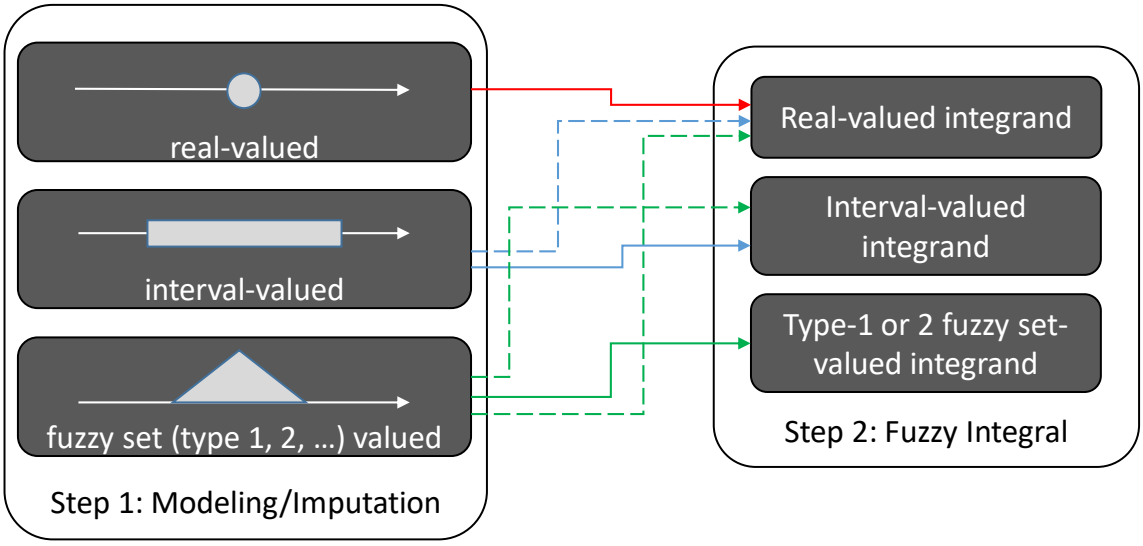



Figure 3.2
Illustration of *connections* (pathways) between modeling/imputation choices and FI extensions. Dashed lines indicate type reduction.

**Scalar-valued ChI:** Equation (3.1) is the simplest and most naive approach considered. The scalar-valued ChI does not take into account uncertainty about missing data. As expected, this action has consequences (elaborated on in Example 3).

**Example 3. (Tendency of modeled/imputed data to dictate the ChI)** For sake of analysis and without loss of generality, we consider that all observed values lie within $[0, 1]$, i.e., $o_j(x_i) \in [0, 1], \forall i, j$. Let the default imputed value be $c$, i.e., $o_j(x_i) = c,\ \forall j,\ x_i \in X^h, c \in [0, 1]$. It is trivial to show that this method runs the risk of controlling/determining the ChI. For example, consider a maximum ChI, i.e., $\mu_1(A) = 1, \forall A \subseteq X \setminus \emptyset$ and a minimum ChI, i.e., $\mu_2(A) = 0, \forall A \subset X$. If $c = 1$ then $C_{\mu_1}(o_j) = 1$ regardless of what is observed. Conversely, if $c = 0$ then $C_{\mu_2}(o_j) = 0$. While these are extreme cases, they illustrate the fact that imputed data is important. If the value $c$ is a default constant with little-to-no meaning, then the output of the ChI is questionable, to say the least. For example, if we let $c = 0.5$ to express "ignorance" and all observed values are higher than $0.5$ then $\mu_2$ will always yield $0.5$ (same argument holds for all observed values less than $0.5$ and $\mu_1$). In reality, we have uncertainty and our result does not show that.

**Distribution-to-scalar ChI:** A logical next step is to type reduce uncertain data for use in Equation (3.1). For example, if a normal distribution is modeling historic and/or current data then the first moment, the mean, can be used. Semantically, our fusion result now reflects a combination of what we observed and the expected value of missing data. Whereas we improved the modeling of missing data we still sadly *loose* information when we type reduced for the ChI.

**Interval-valued ChI:** Next, we outline how to use interval-valued uncertainty about missing data in the ChI. Regardless of how the uncertainty is obtained, e.g., type-reduction

of a normal distribution into a second moment interval, $\bar{o}_j(x_k) = [m_{x_k} - \beta\sigma_{x_k}, m_{x_k} + \beta\sigma_{x_k}]$, the interval-valued ChI [8],

$$\bar{C}_\mu(\mathbf{o}_j) = [C_\mu(\mathbf{o}_j^-), C_\mu(\mathbf{o}_j^+)], \tag{3.7}$$

is two $\Re$-valued ChIs, one integral on the interval left endpoints and a different integral on the interval right endpoints.

**Remark 3. (Trapezoidal Membership Functions)** Consider the trapezoidal membership function, specified by four $\Re$-valued numbers, $a \leq b \leq c \leq d$. It is well-known that a trapezoidal membership function is fully characterized by two level (alpha) cuts, one at $0$, associated with $[a, d]$ and one at $1$ associated with $[b, c]$. First, we level cut each set. Next, we compute the ChI at each level. Last, we type increase the ChI intervals back into a trapezoidal membership function. Note, there was no information loss. It is trivial to prove this as the ChI is monotonic and the trapezoidal values at each level cut between $0$ and $1$ are linear equations.

**Type-1 and type-2 fuzzy set-valued ChIs**: In [194], we put forth two ChI extensions, the gFI and NDFI, for unrestricted (potentially subnormal and non-convex) fuzzy set-valued integrands. In [85], we put forth a ChI extension for type-2 fuzzy sets. There is not sufficient space to review these extensions. The reader can see [194, 85] for mathematical and algorithmic details. The point is, if missing data is modeled by fuzzy sets then there are ChI extensions that can handle this task.

**Remark 4. (Amount of missing data)** The previous sections are focused on modeling, imputation and ChI computation. In Example 3 it was observed that the answer can be

drastically skewed by modeled or imputed data. The question investigated here is, what is the impact of different amounts of missing data? It turns out that this is difficult-to-impossible to solve as the answer depends on the missing data, but more importantly the selection of aggregation operator (underlying capacity). Meaning, if an *extreme* operator such as maximum or minimum (t-conorm (union) and t-norm (intersection) respectively) is selected then one poorly modeled or imputed data can have a catastrophic impact, regardless of how much data is missing. On the other hand, a robust operator like the expected value (e.g., mean, median, etc.) can be expected to perform better–of course degrading with respect to amount of missing data. This holds regardless of modeling/imputation and subsequent ChI computation method. For example, Example 3 naturally extends to the interval-valued ChI (two $\Re$-valued ChIs) and fuzzy set-valued ChI (which is decomposable to interval ChIs and then $\Re$-valued ChIs). The point of this remark is not mathematical characterization but observation that we need to select our aggregation operator with care for missing data.

### 3.4 Missing Data-Driven ChI Learning

In this section, a data-driven method is put forth for observed data only ChI learning. In [12], we put forth a data-driven QP and regularization (for minimal model complexity) approach to ChI learning. However, like all other ChI learning algorithms to date it was assumed that no data is missing. Here, we outline a way to learn the ChI with respect to a set of training data with missing inputs.

Since OP1 is expressed in terms of the full lexicographically encoded $\mu$, one possibility is to represent $\mu_j^w$ in terms of $\mu$ to facilitate optimization of a common set of variables. First, we represent each $\Delta$ as a factor of $1 - \mu(X_j^w)$,

$$\Delta\mu_j^w(S_{\pi_j^w(i)}) = \alpha(S_{\pi_j^w(i)})(1 - \mu(X_j^w)),$$

where $\alpha(S_{\pi_j^w(1)}) = 1$ and $0 \leq \alpha(S_{\pi_j^w(i)}) \leq 1,\ i = 2, \ldots, N_j^w$. The parameter $\alpha(S_{\pi_j^w(i)})$ is user defined and it can be selected per layer or node in the FM[1]. Note, we already discussed three applicable strategies; e.g., upper boundary fix, $\alpha(S_{\pi_j^w(1)}) = 1$ and $\alpha(S_{\pi_j^w(i)}) = 0$ for $i = 2, \ldots, N_j^w$. The ChI with respect to $\mu_j^w$, in terms of $\mu$, is

$$\begin{aligned}
C_{\mu_j^w}(\mathbf{o}_j) &= \sum_{i=1}^{N_j^w}[o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})][\mu(S_{\pi_j^w(i)}) \\
&+ \alpha(S_{\pi_j^w(i)})(1 - \mu(X_j^w))] \\
&= \sum_{i=2}^{N_j^w}[o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})]\mu(S_{\pi_j^w(i)}) + \\
&\sum_{i=1}^{N_j^w}[o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})]\alpha(S_{\pi_j^w(i)}) - \sum_{i=2}^{N_j^w}[o_j(x_{\pi_j^w(i)}) \\
&- \mu(X_j^w)o_j(x_{\pi_j^w(i-1)})]\alpha(S_{\pi_j^w(i)}),
\end{aligned}$$

which can be expressed in matrix form as

$$C_{\mu_j^w}(\mathbf{o}_j) = \mathbf{c}_j^{wT}\mathbf{u} + b, \tag{3.8}$$

[1] Alternatively, the scaling function $\alpha$ can be learned in conjunction with the FM, possibly via alternating optimization (subject of future work).

where the coefficient $c_{jk}^w$ for the $k$th variable, $\mu(B)$, is $o_j(x_{\pi_j^w(l)}) - o_j(x_{\pi_j^w(l-1)})$ if $\exists S_{\pi_j^w}(l) = B, l \in \{2, \ldots, N_j^w\}$, $-\sum_{i=2}^{N_j^w}[o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})]\alpha(S_{\pi_j^w(i)})$, if $S_{\pi_j^w}(1) = B$, 0 otherwise and $b = \sum_{i=1}^{N_j^w}[o_j(x_{\pi_j^w(i)}) - o_j(x_{\pi_j^w(i-1)})]\alpha(S_{\pi_j^w(i)})$. The SSE for the data is therefore

$$E_2(O, \mathbf{u}) = \sum_{j=1}^{M}(\mathbf{c}_j^{wT}\mathbf{u} + b - y_j)^2 = \sum_{j=1}^{M}(\mathbf{c}_j^{wT}\mathbf{u} - y_j^w)^2,$$

where $y_j^w = y_j - b$. This SSE equation has the same form as Equation (3.3) and therefore OP1 can be used to learn the FM – $\mathbf{c}_j$ and $y_j$ need to be replaced by $\mathbf{c}_j^w$ and $y_j^w$ respectively. Equation (3.8) can be used to compute the ChI of data with respect to $\mu$ and scaling coefficients $\alpha(S_{\pi_j^w(i)})$.

## 3.5 Case Study and Synthetic Experiment

The previous sections are focused on concepts and methods for fusing missing data with the FI. This section explores these ideas via a case study and synthetic experiment. Energy is not expended on proving that the FI is a useful tool for any one application, e.g., sensor data fusion, computer vision, multi-criteria decision making, etc., this has already been well-established by the field.

### 3.5.1 Case Study from Forensic Anthropology

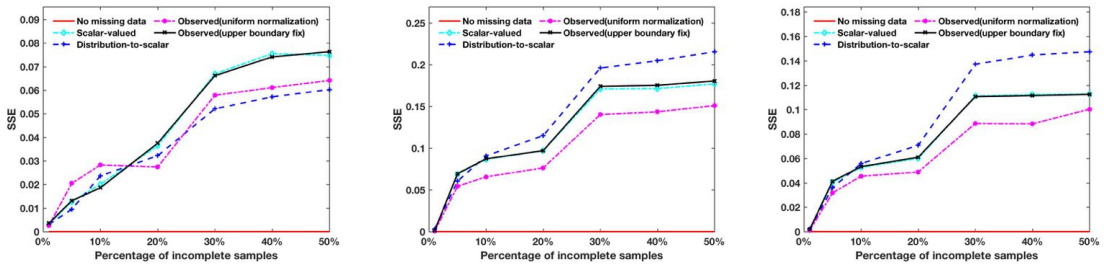
In this subsection, we discuss fusion for skeletal-age-at-death estimation in forensic anthropology. In [14, 8, 7, 10, 6], the task is to determine the age that an individual died, via natural or by active intervention (unnatural death), based on recovered skeletal remains and established aging methods. Different bones are usually present–making it a missing data problem. Sadly, this was not recognized, at a minimum appreciated, until now. For each

bone and aging method, we obtain an input fuzzy set defined on the "age domain" (e.g., $[0, 120]$) where the membership degree is the support in age-at-death. For example, we might have evidence of age-at-death based on the Pubic Symphysis, Auricular Surface and Ectocranial Valt Suture Closure (three inputs from three different bones and aging methods, e.g., Todd $1920$, Lovejoy et al. $1985$b and Meindl and Lovejoy $1985$). Input one might tell us $[25, 26]$, input two might say $[25, 29]$ while input three says $[24, 60]$. Along with each input we have a *bone quality*. We assign a score of $\{1.0, 0.8, 0.6, 0.4, 0.2, 0.1, 0.0\}$ based on the weathering of the bone (corresponding to Stages $0$ to $6$ for a modified version of Behrensmeyer weathering stages provided in Standards). The result is typically a subnormal-valued convex fuzzy set. In addition, we use known accuracies (correlation coefficients) for each aging method. These values are the FM singleton variables (aka densities). As we did not have access to the full FM, the Sugeno $\lambda$ FM [175] (a well-known imputation method) was used to assign the remaining FM variables relative to the densities for observed data.

This forensic application is discussed herein because it's an interesting example of the observation only FI. In our current article, one idea is to sample the relevant subset of the FM and normalize it. However, in our forensic application we only have access to the densities–which could be seen as a missing "data" challenge for $\mu$. As the two and above tuples are unknown, an imputation algorithm (the Sugeno $\lambda$ formula) is used to assign FM variable values. If we had privilege to the full FM then we would follow the procedure outlined in this article. In [146] Nguyen et al. showed that the Sugeno $\lambda$ FM is, from a purely mathematical standpoint, equivalent to a probability (i.e., additive)

measure. There is a computational advantage to using the Sugeno $\lambda$ FM versus reducing it into its equivalent probability measure. Specifically, they showed that we can perceive the Sugeno $\lambda$ FM as a re-scaling of the underlying probability measure. The point is, our prior forensic work addressed missing data for both $h$ (observation only FI) and $\mu$ (an additive measure-based imputation formula that is monotonic and normality preserving).

Last, in many cases a remain (e.g., skull) is missing. Previously, we discarded such inputs. Based on the current article, we plan to next explore the use of interval and fuzzy set-valued modeling and imputation forensic algorithms for missing data. Whereas complete ignorance, e.g., $[0, 120]$, might prove to be too extreme, if recovered remains allow us to narrow the range of other missing data then we would like to explore the benefits of computing with respect to uncertainty. At the moment, our current approach is the observation only FI, which is subject to the limitation discussed in Remark 2.

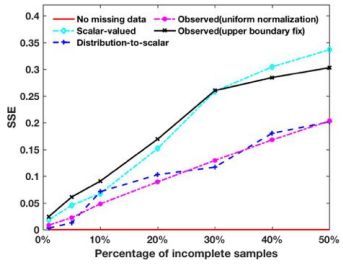



(a) SSE of learned FM and ground truth FM

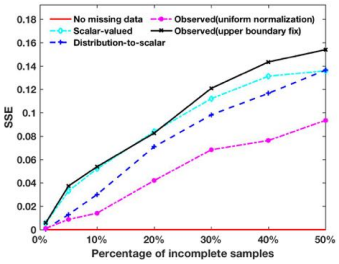
(b) SSE for training data

(c) SSE for test data

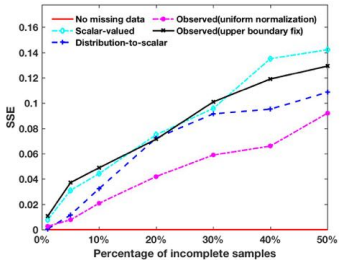
Figure 3.3
Results for soft-max aggregation operator.




(a) SSE of learned FM and ground truth FM

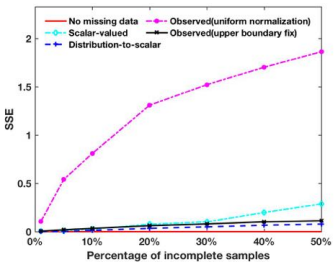



(b) SSE for training data

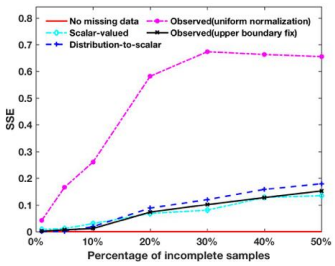



(c) SSE for test data

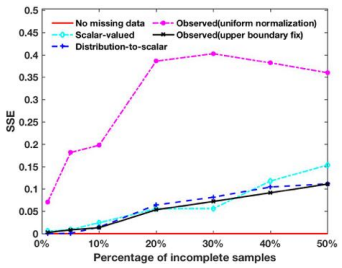
Figure 3.4
Results for mean aggregation operator.




(a) SSE of learned FM and ground truth FM




(b) SSE for training data




(c) SSE for test data

Figure 3.5
Results for soft-min aggregation operator.

### 3.5.2 Synthetic Example: Data-Driven ChI Learning

In this subsection, we conduct a synthetic experiment for data-driven ChI learning. A controlled experiments is preferred to "real data" because we can better investigate benefits and drawbacks related to missing data versus limiting ourself to the "scope" of available data or accuracy results for a process that we do not truly know the underlying answer for. We consider three *ordered weighted averages* (OWAs)[201], which are just *linear combinations of order statistics* (LCOS) when the inputs and weights are $\Re$-valued, which the ChI can generate. The three OWAs are soft-max (t-conorm, union like operator), mean (expected value operator) and soft-min (t-norm, intersection like operator). These three

operators are helpful because they cover the "spectrum" of commonly encountered aggregation philosophies (optimistic, expected value and pessimistic). We let $N = 5$, $M = 150$ and the OWA weights are $(0.71, 0.155, 0.077, 0.039, 0.0019)$ for soft-max, $\frac{1}{5}$ for mean and $(0.0019, 0.039, 0.077, 0.155, 0.71)$ for soft-max,. The data was (pseudo)randomly generated from a truncated normal distribution in $[0, 1]$ with means $[0.50\ 0.17\ 0.33\ 0.67\ 0.83]$ and standard deviation $0.45$. The performance of the different proposed methods are evaluated for different percentages of missing data, $[1\%\ 5\%\ 10\%\ 20\%\ 30\%\ 40\%\ 50\%]$. One input was removed from each sample marked as missing. Three-fold cross-validation is used. The average SSE for the FM to the ground truth FM and also the SSE on the training and test data are used as performance metrics.

A value of $0.5$ is selected for scalar-valued imputation–as it debatably best represents ignorance, $0$ being no support and $1$ being full support. For distribution-to-scalar, we modeled data using a normal distribution and it is type reduced to the mean. Last, both uniform normalization and upper boundary fix are explored for the observation value only FI. Figures (3.3), (3.4), and (3.5) are experiment results. Before we delve into specifics, we highlight that different aggregation philosophies (i.e., selections of FM) result in different trends and no global pattern is present (as expected). This is further elaborated on at the end of the subsection.

Figure (3.3) are the results for soft-max (optimistic aggregation). Overall, the observed-value method (the "safe approach") with uniform normalization leads in each performance criteria. While distribution-to-scalar does well with respect to modeling the true underlying FM, its training and prediction SEE is the worst. Observed-value with upper boundary fix

has relatively poor performance for soft-max as imputation depends on the minimum of the observed values. On the other hand, uniform scaling puts more weight on the higher observed values, yielding lower error for soft-max.

Results for the mean aggregation operator are reported in Figure (3.4). Overall, observed data only with uniform normalization is best again. However, distribution-to-scalar is now second best (versus worst). This is expected as we are using the means of the missing data distributions. Note, observed-value with uniform normalization considers only the observed inputs. As such, its scaling of the FM terms is equivalent to imputing missing values proportional to the FM of the observed sources. Therefore, the imputed value in mean FM is proportional to the sum (or mean) of the observed values.

Figure (3.5) are the results for soft-min (pessimistic operator). Whereas observed data with uniform normalization was best before, it is by far the worst here. The reason is, in soft-min the top node of the sub-lattice is relatively small and therefore the FM values are scaled by a larger factor than in soft-max–making the normalized sub-FM closer to a FM of all ones (which is the maximum operator for the ChI). Distribution-to-scalar and observed data with upper boundary fix provide better results, where the imputation of the latter depends on the minimum of the observed values, which partially helps to preserve the aggregation behavior.

Overall, as stated in the start of this subsection, the most interesting takeaway is the performance variation across aggregation *philosophies* (minimum, average and maximum). There is no global winner across percentages of missing data and aggregations, which is expected based on the theory and remarks in previous sections. This is interpreted as fol-

lows. Missing data is not a simple problem. Care should be given with respect to studying all that is possible for for a task at hand. The next step is to explore the different options, compute with respect to just observed data or model/impute, and it all should be done relative to underlying aggregation operator (selection of FM).

### 3.6 Conclusion and Future Work

Herein, the question of how to extend the FI to problems involving missing data was investigated. We identified two paths, use just observed data or fuse modeled/imputed data with observed data. Three strategies were put forth to make the ChI work with respect to observed data and a complete FM. Furthermore, different modeling and FI extensions were discussed for the inclusion and fusion with respect to missing data. It was shown that there is no "winning method". Instead, the different choices are dictated by the properties of an application (context). It was shown that whereas the observation-valued ChI is a safe route, it runs the risk of not including uncertainty about our missing data. However, while modeled/imputed data leads to a more informed result, if we are not careful about what aggregation operator we use then missing data often dictates, in a potentially destructive fashion, the result. Last, a data-driven algorithm was presented for historic and/or other current observed data.

This article is just a first step towards missing data and the FI. In future work, we will expand our scope to include both missing integrand ($h$) and missing FM ($\mu$), the latter only slightly touched on in our case study. We will also attempt to simultaneously solve for the FM in conjunction with how to normalize it (versus specify the normalization). We will

also explore pathways involving "up sampling" (type increasing). We also want to provide a field guide for practitioners so a user can connect their applications properties to the best set of modeling, imputation and FI extension choices. Next, we put forth an extension for data driven learning. In future work we will study interval and fuzzy set valued label driven learning, or learning of such data in light of scalar-valued inputs. Last, we observed that selection of aggregation operator has a big impact, to say the least. In future work a goal will be to connect different missing data choices to the most appropriate aggregation operator (i.e., underlying FM).

# CHAPTER IV

# VISUALIZATION AND LEARNING OF THE CHOQUET INTEGRAL WITH LIMITED TRAINING DATA

## 4.1 Introduction

In many fields, we are often faced with the task of making decisions based on a set of feature-vector data $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$. This data is typically accompanied by a set of training labels for each feature-vector, giving the pair $(\mathbf{y}, X)$, where $\mathbf{y} = (y_1, y_2, ..., y_n)^T$ is a vector of labels such that $y_i$ is the label of feature-vector $\mathbf{x}_i$. This problem can be considered a classification task, and is typically tackled by training a classifier such that it can accurately predict the class label of a new sample of data where the label is not known. More concretely, the data $(\mathbf{y}, X)$ are used to learn some prediction function $f$ such that we can accurately predict the label of feature vectors as $y = f(\mathbf{x})$.

Linear classifiers are typically nothing more than a hyperplane in the feature-space representing the decision boundary, and training these classifiers involves finding the hyperplane's parameters in some optimal way. A very popular hyperplane classifier is the *support vector machine* (SVM) because it is easy to train and computationally efficient. The drawback to linear SVMs (and other linear classifiers), however, is that they require the data to be linearly separable—a distribution very rarely encountered with real data. One way around this is to instead use their kernel-based variants where the data are non-

linearly projected to a high-dimensional space where a suitable hyperplane is more likely to be found. While this appears to solve the problem of non-separable data, it has its own baggage: what kernel function should be used?

*Multiple kernel learning* (MKL) typically answers this question by learning a new kernel through the combination of predetermined kernels while maintaining symmetry and positive-semidefiniteness, an approach discussed in many works [124, 200, 49, 97, 98, 152, 150]. These approaches fall under the roof of feature-level fusion in that they combine different "looks" at the data (each represented by an individual kernel) and use a single classifier to determine the predicted class label. Another MKL technique uses multiple kernel-based classifiers, each utilizing a different kernel. The outputs of these classifiers is then combined at the decision-level using some aggregation function. This approach to decision-level fusion is the premise for the *decision-level fuzzy integral multiple kernel learning* (DeFIMKL) classifier discussed in Section 4.3, where aggregation is performed via the Choquet *fuzzy integral* (FI) with respect to a *fuzzy measure* (FM). Once again though we have a roadblock: how do we specify the FM?

The task we investigate in this work is learning a FM. Many previous works [75, 11, 26] have shown that an underlying FM can be learned from training data, though here we show that only a subset of the FM is accurately learned from the training data and the remaining FM terms simply follow the constraints from the learning process. In other words, only a subset of the FM is learned in a data-driven manner. Thus when asked to classify a new sample of data using the Choquet FI, we risk utilizing terms from the FM that were not learned accurately from the training data, leading to an erroneous prediction. In this work,

we propose a method to more accurately learn the FM terms that are not data-driven. The method assumes that some knowledge of the underlying FM structure is known and thus can be encoded in the learning process as discussed in Section 4.4.

The remainder of this paper is organized as follows. Section 4.2 discusses fuzzy measures and the Choquet fuzzy integral; it also introduces our strategy of simultaneously visualizing the FM and behavior of the Choquet integral. Section 4.3 reviews learning a fuzzy measure through minimizing the *sum-of-squared error* (SSE) via *quadratic programming* (QP)—the backbone of the DeFIMKL algorithm—as well as its behavior with insufficient training data. Section 4.4 proposes an extension to the DeFIMKL algorithm, allowing knowledge of the underlying FM to be encoded into the QP, and Section 4.5 summarizes experiments with real-world and contrived datasets. Finally, Section 4.6 concludes the paper and discusses our future work.

## 4.2 Fuzzy Measures and Fuzzy Integrals

FIs and FMs are used for many applications and for many types of data, from simple numeric data to intervals and type-2 fuzzy sets [8, 192, 76, 7]. While manual specification of the FM works for small sets of sources, manually specifying the values of the FM for large collections of sources is virtually impossible. Thus, automatic methods have been proposed, such as the Sugeno $\lambda$-measure [73] and the $S$-decomposable measure [56], which build the measure from the densities[1], and genetic algorithm [14, 98], Gibbs sampling [136] and other learning methods, which build the measure by using training data.

[1]The FM values of the singletons, $g(\{x_i\}) = g^i$ are commonly called the *densities*.

Other works [193, 87, 86] have proposed learning FMs that reflect trends in the data and have been specifically applied to crowd-sourcing, where the worth of individuals is not known, and is thus extracted from the data.

### 4.2.1 Fuzzy measures

A measurable space is the tuple $(X, \Omega)$, where $X$ is a set and $\Omega$ is an $\Omega$-algebra or set of subsets of $X$ such that

P1. $X \in \Omega$;

P2. For $A \subseteq X$, if $A \in \Omega$, then $A^c \in \Omega$;

P3. If $\forall A_i \in \Omega$, then $\bigcup_{i=1}^{\infty} A_i \in \Omega$.

A FM is a set-valued function, $g : \Omega \to [0, 1]$, with the following properties:

P4. (Boundary conditions) $g(\emptyset) = 0$ and $g(X) = 1$;

P5. (Monotonicity) If $A, B \in \Omega$ and $A \subseteq B$, $g(A) \leq g(B)$.

If $\Omega$ is an infinite set, then there is also a third property to guarantee continuity; however, in practice and in this paper, $\Omega$ is finite and thus this property is unnecessary. While fuzzy measures provide a way for quantifying the worth of combinations of sources, fuzzy integrals can be used to aggregate the information from these sources.

### 4.2.2 Fuzzy integrals

There are many forms of the FI; see [73] for detailed discussion. In practice, FIs are frequently used for evidence fusion [72, 117, 18, 14]. They combine sources of information

by accounting for both the support of the question (the evidence) and the expected worth of each subset of sources (as supplied by the FM $g$). Here, we focus on the fuzzy Choquet integral, proposed by Murofushi and Sugeno [44, 143]. Let $h : X \to \mathbb{R}$ be a real-valued function that represents the evidence or support of a particular hypothesis.[2] The discrete (finite $\Omega$) fuzzy Choquet integral is defined as

$$\int_C h \circ g = C_g(h) = \sum_{i=1}^{n} h(x_{\pi(i)}) \left[ g(A_i) - g(A_{i-1}) \right], \tag{4.1}$$

where $\pi$ is a permutation of $X$, such that $h(x_{\pi(1)}) \geq h(x_{\pi(2)}) \geq \ldots \geq h(x_{\pi(n)})$, $A_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$, and $g(A_0) = 0$ [76, 176]. Detailed treatments of the properties of FIs can be found in [76, 69, 176].

### 4.2.3 Common Aggregations via the Choquet Integral

It is well known that the Choquet integral is a powerful aggregation operator parametrized by a FM, and thus can represent many aggregation functions [181]. For example, the Choquet integral acts as the maximum operator when the FM is all 1s (except $g\{\emptyset\} = 0$, due to boundary constraints), the minimum operator when the FM is all 0s (except $g\{X\} = 1$, due to boundary constraints), and the mean operator when $g(A_i) = |A_i|/n, \forall A_i \subset X$.

### 4.2.4 Visualizing the Fuzzy Integral

The FM lattice (Hasse diagram) is a convenient method to visualize a FM; Figure 4.1 illustrates the lattice of a FM for the case of $n = 3$. Note that the size of the individual

[2] Generally, when dealing with information fusion problems it is convenient to have $h : X \to [0, 1]$, where each source is normalized to the unit-interval.

nodes in the lattice indicates their relative magnitude, and monotonicity is apparent since nodes at higher levels in the lattice are larger—or at least as large—than those below.

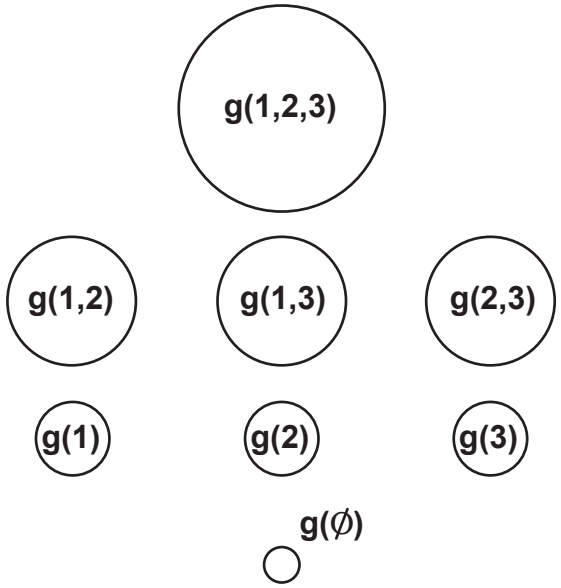
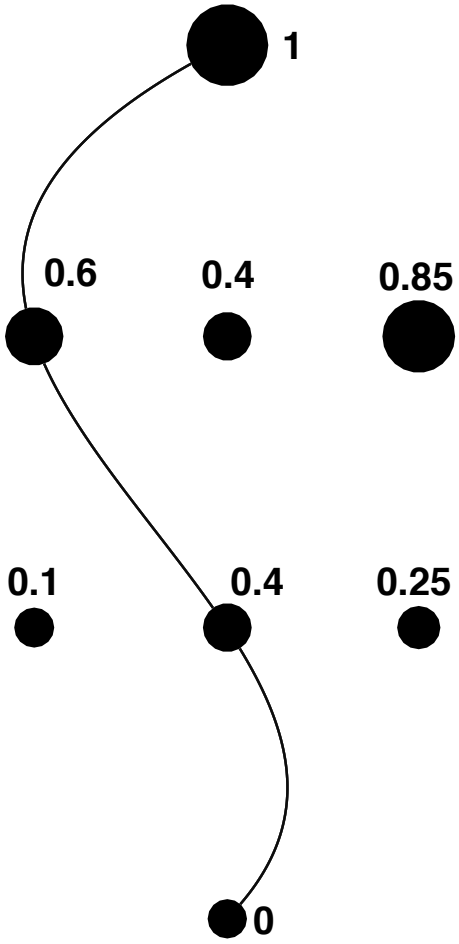
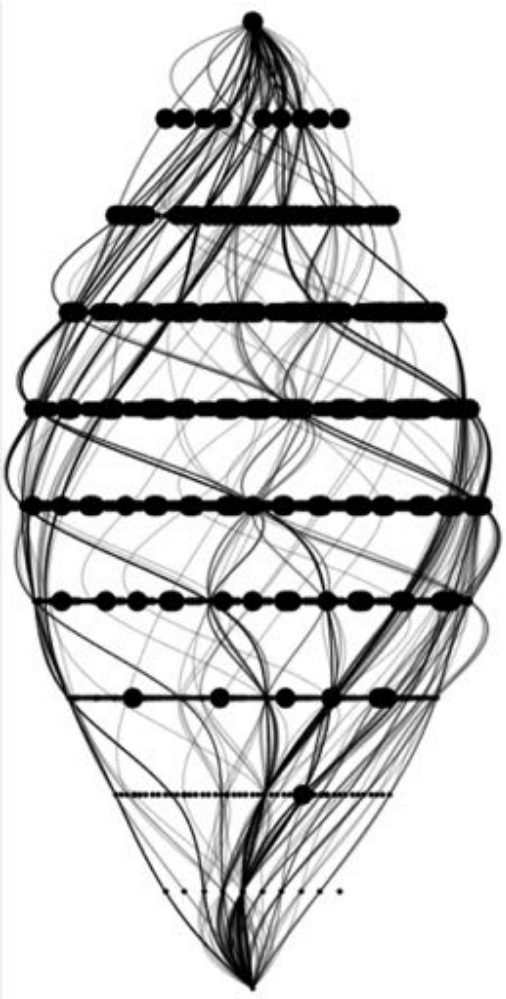



Figure 4.1
Lattice of FM elements for $n = 3$. Monotonicity (P5) is illustrated by the size of each node, i.e., $g(\{x_1\}) \leq g(\{x_1, x_2\})$ as $\{x_1\} \subset \{x_1, x_2\}$. Note that shorthand notation is used where $g(1, 3)$ is equivalent to $g(\{x_1, x_3\})$.

The FM lattice alone, while useful for showing a FM, does not give insight into how the Choquet integral at (4.1) utilizes the lattice due to the $\pi$-permutation. Therefore, for a particular input we also show the path through the lattice followed by the Choquet integral. For example, suppose that a particular data sample $x$ and hypothesis $h$ gives rise to the permutation $\pi = \{2, 1, 3\}$. Then, for an arbitrary FM, the lattice visualization includes the path shown in Figure 4.2(a). This visualization strategy allows us to summarize the FM as well as the Choquet integral's paths.




(a) The path taken by the Choquet integral due to a single input inducing the permutation $\pi = \{2, 1, 3\}$. Note that the FM was arbitrarily defined in this example, and their distribution (ordering) follows that of Figure 4.1.



(b) Lattice of learned FM and paths for random training data from the Ionosphere data set using $m = 10$. Note there are numerous untouched nodes and their learned values are driven by the constraints in (4.9).

Figure 4.2
Lattice visualization examples.

## 4.3 The DeFIMKL Algorithm

The DeFIMKL algorithm was introduced in [150] as a method of decision-level fusion in the context of classification, where a set of decisions from an ensemble of classifiers are non-linearly fused via the Choquet FI. To mathematically describe the algorithm, let the decision-value for feature-vector $\mathbf{x}_i$ from the $k$th classifier in an ensemble be $f_k(\mathbf{x}_i)$; the set of decisions from the ensemble comprise the evidence $h$ for the Choquet integral. The evidence is then integrated with respect to the FM $g$, which encodes the relative worth of each classifier in the ensemble. This results in the ensemble decision $f_g(\mathbf{x}_i)$ for feature-vector $\mathbf{x}_i$ with respect to the FM $g$,

$$f_g(\mathbf{x}_i) = \sum_{k=1}^{m} f_{\pi(k)}(\mathbf{x}_i) \left[ g(A_k) - g(A_{k-1}) \right], \tag{4.2}$$

where $A_k = \{f_{\pi(1)}(\mathbf{x}_i), \ldots, f_{\pi(k)}(\mathbf{x}_i)\}$, such that $f_{\pi(1)}(\mathbf{x}_i) \geq f_{\pi(2)}(\mathbf{x}_i) \geq \ldots \geq f_{\pi(m)}(\mathbf{x}_i)$. This method has been explored in many previous works as a generalized classifier fusion method [18, 181, 195, 212].

The FM completely specifies the behavior of the Choquet integral. Thus, the next step in understanding the DeFIMKL algorithm is assigning a FM for the Choquet integral in (4.2), of which there are many methods. For example, the Sugeno $\lambda$-measure [73] may be naively used after specifying the FM values of the singletons; however, there is no guarantee that this choice of FM will yield acceptable results when used with (4.2) since it does not take training data into account. To address this, we suggested a data-driven method to learn the FM $g$ through regularized *sum-of-squared error* (SSE) optimization in [12]. This method is summarized next.

Let the SSE be defined as

$$E^2 = \sum_{i=1}^{n} \left( f_g(\mathbf{x}_i) - y_i \right)^2 . \tag{4.3}$$

It can be shown that (4.2), as a Choquet integral, can be reformulated as

$$f_g(\mathbf{x}_i) = \sum_{k=1}^{m} \left[ f_{\pi(k)}(\mathbf{x}_i) - f_{\pi(k+1)}(\mathbf{x}_i) \right] g(A_k), \tag{4.4}$$

where $f_{\pi(m+1)} = 0$ [176]. We can then expand the SSE as

$$E^2 = \sum_{i=1}^{n} \left( H_{\mathbf{x}_i}^T \mathbf{u} - y_i \right)^2 , \tag{4.5a}$$

where $\mathbf{u}$ is the lexicographically ordered FM $g$, i.e., $\mathbf{u} = (g(\{x_1\}), g(\{x_2\}), \ldots, g(\{x_1, x_2\}),$ $g(\{x_1, x_3\}), \ldots, g(\{x_1, x_2, \ldots, x_m\}))$, and

$$H_{\mathbf{x}_i} = \begin{pmatrix} \vdots \\ f_{\pi(1)}(\mathbf{x}_i) - f_{\pi(2)}(\mathbf{x}_i) \\ \vdots \\ 0 \\ \vdots \\ f_{\pi(m)}(\mathbf{x}_i) - 0 \end{pmatrix}, \tag{4.5b}$$

where $H_{\mathbf{x}_i}$ is of size $(2^m - 1) \times 1$ and contains all the difference terms $f_{\pi(k)}(\mathbf{x}_i) - f_{\pi(k+1)}(\mathbf{x}_i)$ at the corresponding locations of $A_k$ in $\mathbf{u}$. Finally, folding out the squared term in (4.5a) produces

$$\begin{aligned} E^2 &= \sum_{i=1}^{n} \left( \mathbf{u}^T H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T \mathbf{u} - 2 y_i H_{\mathbf{x}_i}^T \mathbf{u} + y_i^2 \right) \\ &= \mathbf{u}^T D \mathbf{u} + \mathbf{f}^T \mathbf{u} + \sum_{i=1}^{n} y_i^2, \\ D &= \sum_{i=1}^{n} H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T, \quad \mathbf{f} = -\sum_{i=1}^{n} 2 y_i H_{\mathbf{x}_i}. \end{aligned} \tag{4.6}$$

Since (4.6) is a quadratic function, we can add constraints on $\mathbf{u}$ such that it represents a FM, leading to a constrained QP. We can write the boundary and monotonicity constraints on $\mathbf{u}$ (see properties P4 and P5) as $C\mathbf{u} \leq 0$, where

$$C = \begin{pmatrix} \Psi_1^T \\ \Psi_2^T \\ \vdots \\ \Psi_{n+1}^T \\ \vdots \\ \Psi_{m(2^{m-1}-1)}^T \end{pmatrix} \tag{4.7}$$

and $\Psi_1^T$ is a vector representation of the monotonicity constraint, $g\{x_1\} - g\{x_1, x_2\} \leq 0$. Hence, $C$ is simply a matrix of $\{0, 1, -1\}$ values of size $(m(2^{m-1} - 1)) \times (2^m - 1)$ with the form

$$C = \begin{bmatrix} 1 & 0 & \cdots & -1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & -1 \end{bmatrix}. \tag{4.8}$$

Thus, the full QP to learn the FM $\mathbf{u}$ is

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u}, \quad C\mathbf{u} \leq \mathbf{0}, \quad (\mathbf{0}, 1)^T \leq \mathbf{u} \leq \mathbf{1}, \tag{4.9}$$

where $\hat{D} = 2D$. Note that an additional regularization term can be included in the QP as

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda v_*(\mathbf{u}), \tag{4.10}$$

where $\lambda$ is the regularization weight and $v_*(\cdot)$ is some regularization function. For example, $\ell_p$-norm regularization is applied when $v_*(\mathbf{u}) = \|\mathbf{u}\|_p$. $\ell_1$ and $\ell_2$ regularization of this QP are discussed in [12, 150].

The QPs at (4.9) and (4.10) provide a method to learn the FM $\mathbf{u}$ (i.e., $g$) from training data, thus completing the requirements for calculating the Choquet integral at (4.2). We now review how to use a kernel classifier to determine the decision-value $f_k(\mathbf{x}_i)$. Specifically, we will show how to use the SVM with this algorithm.

Suppose that each learner $f_k(\mathbf{x}_i)$ is a kernel SVM, each trained on a separate kernel $K_k$. The SVM classifier decision value is

$$\eta_k(\mathbf{x}) = \sum_{i=1}^{n} \alpha_{ik} y_i \kappa_k(\mathbf{x}_i, \mathbf{x}) - b_k, \tag{4.11}$$

which is interpreted as the distance of $\mathbf{x}$ from the hyperplane defined by the learned SVM model parameters, $\alpha_{ik}$ and $b_k$ [50, 32]. The class label is typically computed as $\text{sgn}\{\eta_k(\mathbf{x})\}$,[3] which could be used as the training input to the FM learning at (4.6), however, we remap $\eta_k(\mathbf{x})$ onto the interval $[-1, +1]$ via the sigmoid function to create inputs for learning as

$$f_k(\mathbf{x}) = \frac{\eta_k(\mathbf{x})}{\sqrt{1 + \eta_k^2(\mathbf{x})}}. \tag{4.12}$$

Thus, the training data for DeFIMKL are $(\{K_k = [\kappa_k(\mathbf{x}_i, \mathbf{x}_j)], \mathbf{f}_k(X)\}, \mathbf{y}), k = 1, \ldots, m$, where $K_k$ are the kernel matrices for each kernel function $\kappa_k$, $\mathbf{f}_k(X) = (f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_n))^T$ are the remapped SVM decision values, and $\mathbf{y} = (y_1, \ldots, y_n)$ are the ground-truth labels of $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, respectively; the output of the QP learner is the FM $g$. Algorithm 4

[3] Note that the $\text{sgn}(\cdot)$ function discards information about how well the kernel separates the classes of data.

summarizes the training process. After training, a new feature vector $\mathbf{x}$—from a test data set—can be classified by via the procedure summarized in Algorithm 3.

**Algorithm 1:** DeFIMKL Classifier Training

**Data:** $(\mathbf{x}_i, y_i)$ - feature vector and label pairs; $K_k$ - kernel matrices

**Result:** $\mathbf{u}$ - Lexicographically ordered fuzzy measure vector

1 **for** *each kernel matrix* **do**

2 Compute the kernel SVM classifier decision values, $\eta_k$, as in (4.11).

3 Remap the decision values onto the interval $[-1, +1]$ as $f_k$ using (4.12).

4 Solve the minimization problem in (4.9) for the FM $\mathbf{u}$.

**Algorithm 2:** DeFIMKL Classifier Prediction

**Data:** $\mathbf{x}$ - feature vector; $K_k$ - kernel matrices; $\mathbf{u}$ - learned fuzzy measure vector

**Result:** $y$ - Predicted class label

1 Compute the SVM decision values $f_k(\mathbf{x})$ by using (4.11) and (4.12).

2 Apply the Choquet integral at (4.2) with respect to the learned FM $\mathbf{u}$.

3 Compute the class label as $y = \text{sgn}\{f_g(\mathbf{x})\}$.

### 4.3.1 FM Learning Behavior with Insufficient Training Data

Learning the entire FM for a DeFIMKL classifier utilizing $m$ classifiers requires at least $2^m$ (or $2^m - 2$, observing the boundary conditions in property P4) *rank-independent* observations. Therefore, since so many rank-independent observations are rarely encountered

in training data sets, there will likely be values of the FM that are not data-driven. Figure 4.2(b) shows an example of this in the wild, where the Ionosphere dataset [4] was used to train DeFIMKL with 10 classifiers. Note that there are many nodes in the lattice that are never "touched" by the training data; the learned values for these nodes is completely driven by the monotonicity constraints in the QP, the choice of regularization used, and the initialization used in the QP solver. It is therefore highly unlikely that the learned values at these nodes accurately represent the underlying FM, and if Algorithm 3 is applied to a new data point that utilizes one or more of the untouched nodes, prediction accuracy will suffer. The following contrived example demonstrates the behavior of the $\ell_2$-regularized DeFIMKL algorithm with insufficient training data.

**Example 4.** *Learning an Underdetermined FM via $\ell_2$-regularized DeFIMKL.* A three-SVM $\ell_2$-regularized DeFIMKL algorithm (i.e., $m = 3$, however these results are also indicative of the behavior when $m > 3$) is trained with a synthetic dataset that purposefully avoids two nodes in the fuzzy lattice and was generated using the underlying FM shown in Table 4.1; the underlying FM was arbitrarily assigned. The FM learned by the DeFIMKL algorithm is also shown in Table 4.1. Note that two nodes in the lattice, corresponding to $g(\{x_2\})$ and $g(\{x_1, x_2\})$ were not driven by the training data, and thus are essentially driven by the monotonicity constraints.

What we see is that all nodes touched by the training data (i.e., nodes traversed by the Choquet integral) are learned successfully with minimal error (well within $5\%$). However, the two nodes untouched by the training data are assigned values based on monotonic-

[4] Retrieved from UCI Machine Learning Repository. Available online at http://archive.ics.uci.edu/ml

ity constraints. The node corresponding to $g(\{x_2\})$ gets a value of essentially 0, satisfying the monotonicity constraint that $g(\{\emptyset\}) \leq g(\{x_2\}) \leq min(g(\{x_1, x_2\}), g(\{x_2, x_3\}))$, and the node corresponding to $g(\{x_1, x_2\})$ gets a value of $0.14$ to satisfy the constraint $max(g(\{x_1\}), g(\{x_2\})) \leq g(\{x_1, x_2\}) \leq g(\{X\})$. Note that in both of these cases, the learned FM value is essentially the minimum value permitted by the monotonicity constraints. This, as will be shown in the following section, is due to the $\ell_2$-regularization of the DeFIMKL algorithm.

Table 4.1
Underlying and learned FMs (excluding $g(\{\emptyset\})$ and $g(X)$ whose values are 0 and 1, respectively, due to the boundary conditions).

| | | **Regularization** | | |
|---|---|---|---|---|
| **FM Term** | **Underlying** | $\ell_2$ **(min)** | **max** | **mean** |
| $g(\{x_1\})$ | 0.14 | 0.14 | 0.19 | 0.14 |
| $g(\{x_2\})$* | 0.29 | 0.00017 | 0.93 | 0.33 |
| $g(\{x_3\})$ | 0.43 | 0.43 | 0.44 | 0.43 |
| $g(\{x_1, x_2\})$* | 0.57 | 0.14 | 1 | 0.67 |
| $g(\{x_1, x_3\})$ | 0.71 | 0.69 | 0.71 | 0.71 |
| $g(\{x_2, x_3\})$ | 0.86 | 0.83 | 0.93 | 0.86 |

*FM terms marked with an asterisk are not addressed by the training data.

## 4.4 FM Learning with a Goal

The standard DeFIMKL algorithm discussed in the previous section assumes that the structure of the underlying FM is not known, thus no information regarding the underlying

FM is encoded in the QP. If, however, the FM is partially known, the QP at (4.10) should include that information. To this end, we propose the regularization function

$$v_*(\mathbf{u}) = \lambda \|\mathbf{u} - \mathbf{g}\|_p^2, \tag{4.13}$$

where $\mathbf{g}$ represents a goal of what we expect the FM to look like. Including this regularization function in the QP (with $p = 2$) gives

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda \|\mathbf{u} - \mathbf{g}\|_2^2, \tag{4.14}$$

and the QP then also simultaneously minimizes the Euclidean distance between the learned FM $\mathbf{u}$ and the goal $\mathbf{g}$. Expanding the regularization term in (4.14) leads to

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \left(\hat{D} + \lambda I\right) \mathbf{u} + (\mathbf{f} - 2\lambda\mathbf{g})^T \mathbf{u}, \tag{4.15}$$

showing that the inclusion of this regularization function still results in a valid QP, though this comes as no surprise since the regularization function in (4.13) is quadratic in $\mathbf{u}$.

Table 4.2
Classification Accuracy of Various Regularization Functions*

| | **Data Set** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Regularization** | Sonar | Derm | Ecoli | Glass | Toy 3 | Toy 5 | Toy 8 |
| None | **80.5 (5.63)** | 94.3 (2.61) | **97.3 (1.90)** | 91.2 (4.39) | 84.7 (6.48) | **95.0 (3.39)** | **98.4 (1.76)** |
| $\ell_1$ | **78.4 (7.23)** $\lambda = 0.5$ | 89.6 (4.35) $\lambda = 0.5$ | 91.8 (2.79) $\lambda = 5$ | 82.7 (6.77) $\lambda = 0.5$ | 64.4 (7.23) $\lambda = 2.5$ | 91.0 (4.87) $\lambda = 0.5$ | **96.9 (2.74)** $\lambda = 5$ |
| $\ell_2$ (min) | **80.0 (6.72)** $\lambda = 0.5$ | 91.9 (3.09) $\lambda = 0.5$ | 91.8 (2.79) $\lambda = 0.5$ | 85.9 (5.79) $\lambda = 0.5$ | 64.2 (7.05) $\lambda = 2.5$ | 92.4 (3.88) $\lambda = 0.5$ | 91.4 (4.36) $\lambda = 5$ |
| max | 72.0 (7.58) $\lambda = 0.5$ | **97.4 (1.88)** $\lambda = 1.5$ | **97.9 (1.91)** $\lambda = 4.5$ | **94.2 (3.97)** $\lambda = 4.5$ | 88.5 (6.35) $\lambda = 1.5$ | 94.3 (2.84) $\lambda = 2$ | **98.9 (1.61)** $\lambda = 2.5$ |
| mean | **76.6 (7.17)** $\lambda = 0.5$ | **97.7 (1.49)** $\lambda = 3$ | **97.1 (2.14)** $\lambda = 1.5$ | **95.2 (3.07)** $\lambda = 1$ | **94.8 (4.50)** $\lambda = 1.5$ | **96.7 (2.30)** $\lambda = 5$ | **98.4 (1.89)** $\lambda = 1$ |

*Bold indicates best result according to a two-valued $t$-test at a 5% significance level.

### 4.4.1 $\ell_2$-regularization: Minimum Aggregation

It is interesting to note that when $\mathbf{g} = 0$, the regularization function in (4.13) reduces to that of $\ell_p$-norm regularization of the FM vector. This is precisely why the learned FM's untouched nodes of last section's example "default" to lie at the lowest end of their allowable range as shown in Table 4.1—we are essentially forcing the untouched FM values to be as close to zero as possible through our choice of $\ell_2$-norm regularization. Tying this with the aggregation operators discussed in Section 4.2.3, we recognize that when $\mathbf{g} = 0$ we are forcing the Choquet integral to aggregate like the minimum function.

### 4.4.2 Maximum Aggregation

Defining the goal as all 1s causes the untouched nodes to default to the maximum end of their allowable range, tuning the Choquet integral's behavior to that of maximum aggregation (see Section 4.2.3). Rerunning the example in Section 4.3.1 with this goal yields the FM summarized in Table 4.1, where it is obvious that the untouched nodes are assigned the maximum possible value permitted by the monotonicity constraints. Note that in this example the learned FM values for $g(\{x_1\})$ and $g(\{x_2, x_3\})$ have been pushed farther from the underlying FM, though they still lie fairly close. This discrepancy is due to the choice of $\lambda$, which essentially "tunes" where the error is incurred in the QP at (4.14)–a larger value of $\lambda$ will force the learned FM to look like the goal $\mathbf{g}$ despite perturbing the data-driven nodes away from their underlying values. $\lambda$ was arbitrarily set to 1 in these experiments.

### 4.4.3 Mean Aggregation

As a final example, we define the goal of the FM to be that of mean aggregation as explained in Section 4.2.3. Doing so leads to the learned FM shown in Table 4.1. Interestingly, the learned FM at the data-driven nodes is more accurate than that of the previous case of maximum aggregation. We attribute this to the fact that the goal of mean aggregation is more similar to the underlying FM than the goal of maximum aggregation.

## 4.5 Experiments

Experiments were performed using no regularization, $\ell_p-$norm regularization, and the goal-based regularization function in (4.13) with the DeFIMKL algorithm on various datasets from the UCI Machine Learning repository as well as toy datasets generated to purposefully exclude 80% of the training of nodes in an arbitrarily generated fuzzy lattice (three toy datasets were generated using 3, 5, and 8 densities, respectively). Each experiment consists of 100 trials, where in each trial a random partition of 80% of the data is used for training and the remaining data is sequestered for testing; the results we report comprise the mean and standard deviation of classification accuracies. Finally, we vary the regularization parameter, $\lambda$, to explore its effect on classification accuracy and the results with the best $\lambda$s are reported; so, essentially we are comparing the best from each algorithm.

### 4.5.1 Results

Table 4.2 summarizes the results of these experiments. The best algorithms for each dataset are shown in bold font; a two-sample t-test at a 5% significance level is used to

determine the statistically best results—hence, more than one algorithm can be considered as best. In all experiments at least one goal-based regularization function emerges as a top performer. We also find that the max and mean goal-based regularization functions achieve superior results on the *Dermatology* and *Glass* datasets, suggesting that the data define an underlying FM that is most similar to mean or max aggregation. There is no clear trend in the results versus the regularization parameter $\lambda$, and not surprisingly the best selection of $\lambda$ varies based on the dataset used.

## 4.6 Conclusion

This paper first introduced a visualization technique that shows both the FM as well as the Choquet integral's path through the lattice. We also proposed and applied a new regularization function to our previously developed decision-level aggregation algorithm known as DeFIMKL. Including this new regularization function in the DeFIMKL algorithm allows knowledge of an underlying FM to be encoded into the algorithm's training procedure; thus, the user can define a particular goal for the FM before learning. We discussed the application of the new regularization function and demonstrated its behavior using synthetic and real-world datasets where we found that tuning the Choquet integral's behavior to that of max or mean aggregation tended to do best across all datasets.

### 4.6.1 Future Work

The regularization extension proposed in this paper allows knowledge of an underlying FM to be encoded into the learning process of DeFIMKL, though we acknowledge the fact that the underlying FM is typically not known. We also previously demonstrated,

however, that much of the underlying FM can be learned with very little error as long as the learning is data-driven, i.e., cases where there is sufficient training data [150]. Thus, future work is focused on extending the proposed idea of goal-based regularization to goal-based learning of the underlying FM at nodes not utilized by the Choquet integral during training, i.e., values of the underlying FM not attainable from the training data.

# CHAPTER V

# CLODD BASED BAND GROUP SELECTION

## 5.1 Introduction

Hyperspectral imaging is a demonstrated technology for numerous earth and space-borne applications involving tasks such as target detection, invasive species monitoring and precision agriculture. However, hyperspectral imaging suffers from the "curse of dimensionality". Of particular interest is new theory for dimensionality reduction or identification of fewer spectral bands for multispectral versus hyperspectral imaging, typically relative to some specific task, which aids efficient computation, improves classification and lowers system cost. Most techniques can be divided into two broad categories—projection or clustering. Projection techniques require all bands initially (versus feature selection) and they are focused on reducing dimensionality. Approaches include *principal component analysis* (PCA), *Fishers linear discriminant analysis* (FLDA) and *generalized discriminant analysis* (GDA), *random projections* (RP), and kernel extensions. Some methods are unsupervised, e.g., PCA and RP, while others are supervised, e.g., FLDA and GDA. Clustering is unsupervised learning and it can be applied to hyperspectral imagery in a number of ways. While it does not automatically do dimensionality reduction, it helps to identify structure and one can take that information and use it for dimensionality reduction or band group selection. For example, in [133] Martinez et al. used an information measure

to compute dissimilarity between bands and they used hierarchical clustering with Ward's single linkage to produce a minimum variance partitioning of the bands. In [102], Imani and Ghassemain used (hard) $c$-means for supervised band grouping. Martinez's method suffers from the limitations of vanilla hierarchical clustering, e.g., how to pick clusters from the dendogram. Imani and Ghassemain's approach suffers from the limitations of the $c$-means clustering algorithm, e.g., initialization, selection of $c$, and lack of ability compared to "soft" clustering (probabilistic, fuzzy or possibilistic).

Herein, we explore a new band grouping approach based on the *improved visual assessment of clustering tendency* (iVAT) [88]. This approach is well-grounded theoretically, and it produces visual results that an expert or additional clustering algorithm, e.g., *clustering on ordered dissimilarity data* (CLODD) [89], can exploit. Our goal was to identify an algorithm that could reproduce the structure that an expert currently finds and also be useful in the context of classification, which might demand different structure than an expert "sees". A common practice is to use a proximity metric like correlation to measure the similarity between bands. Often, contiguous bands are highly similar and this structure "shows up" if one produces an image of the similarity matrix. The CLODD algorithm analyzes a dissimilarity matrix, e.g., distances between vectors in a data set or bands in hyperspectral imaging, and it automatically finds "block-like" structure. Structure is often found in a proximity matrix according to squares of high-contrast along the matrix diagonal. The CLODD algorithm exploits two properties, "edginess" and "contrast". CLODD obtains contiguous band groups. However, we can automatically identify non-contiguous clusters (band groups) if we re-order the bands according to a method like iVAT. Herein,

we explore both contiguous and non-contiguous band groups and compare their relative performances. On one hand, contiguous is useful if we wish to identify a simpler sensor, however it could very likely be the case that non-contiguous bands share similarity and should be grouped and lead to more of a dimensionality reduction approach. Overall, the "answer" to this question is very task specific. While CLODD and iVAT are naturally unsupervised techniques, we also explore a supervised CLODD and iVAT approach based on the construction of a dissimilarity matrix using the data labels. The following sections describe the proposed approach and results.

## 5.2 Methods

First, the hyperspectral data cube (image) is re-arranged to form a 2D data set (so spatial context is lost) where each row represents a pixel in the image and each column is a band. Let the data set be $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \in \mathbb{R}^{n \times b}$, where $n$ is the number of pixels in the image and $b$ is the number of bands. The label for each pixel, $\mathbf{x}_i$ is $y_i \in \{1, 2, \cdots, L\}$, where $L$ is the number of classes. Figure 5.1 shows the major steps in the proposed approach.

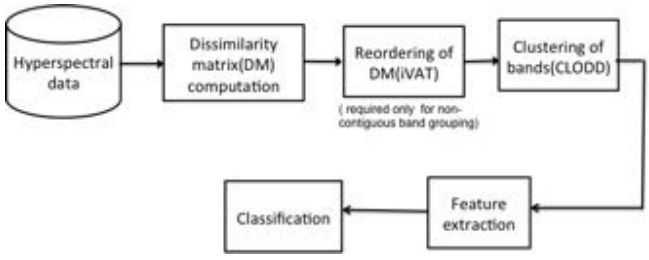



Figure 5.1
Block diagram for the proposed method

Table 5.1
Classification accuracy (percentages) for unsupervised band grouping.

| Bandgrouping method | Feature extraction method | corn (notill) | corn (min) | grass (pasture) | grass (trees) | hay (windowed) | soybeans (notill) | soybeans (min) | soybeans (clean) | woods | Overall of 9 classes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CLODD(contiguous) | Mean | **71.84** | **55.62** | 89.92 | 95.98 | **98.72** | 68.09 | **81.56** | **52.14** | **98.74** | **79.30** |
| CLODD(contiguous) | Weight | 66.43 | 47.98 | 87.41 | 94.97 | **98.72** | 64.86 | 78.98 | 43.18 | 98.65 | 75.95 |
| CLODD(non-contiguous) | Mean | 71.05 | 53.37 | 88.16 | 95.81 | 98.47 | 66.41 | 80.34 | 39.51 | 98.84 | 77.55 |
| CLODD(non-contiguous) | Weight | 66.78 | 47.98 | 87.66 | **96.31** | **98.72** | 65.50 | 78.93 | 41.96 | 98.74 | 76.12 |
| Hierarchical | Mean | 66.87 | 49.48 | 87.91 | 93.13 | 97.70 | 67.05 | 78.88 | 31.57 | 98.45 | 75.39 |
| Hierarchical | Weight | 65.30 | 51.42 | **90.68** | 95.14 | 97.95 | **68.60** | 78.52 | 47.66 | 98.55 | 76.78 |

### 5.2.1 Calculation of dissimilarity matrix

The computation of the *dissimilarity matrix* (DM) differs for unsupervised and supervised band grouping. Note, there are numerous proximity measures and their aggregation that can, and have, been used for each, e.g., correlation, Bhattacharyya distance, Kullback-Liebler divergence, etc [22].

Herein, for supervised band grouping we compute the mean of the training samples in each class and a matrix, $M$, is formed such that $i$th row is the mean vector for the $i$th class. The square of the Euclidean distance between two bands, $i$ and $j$ is computed according to $d_1(i,j) = ||M_i - M_j||^2$, where $M_i$ (and $M_j$) is the $i$th ($j$th respectively) mean vector. For unsupervised band grouping, we compute the square of the $l_2$ norm of the differences between pixel values for those two bands, $d_2(i,j) = ||X_{.,i} - X_{.,j}||^2$, where $X_{.,i}$ is a column vector of all pixel values for band $i$. The resultant pairwise dissimilarity matrix is sized $b \times b$. Figure 5.2 is an example for the Indian Pines data set.

Table 5.2
Classification accuracy (percentages) for supervised band grouping.

| Bandgrouping method | Feature extraction method | corn (notill) | corn (min) | grass (pasture) | grass (trees) | hay (windrowed) | soybeans (notill) | soybeans (min) | soybeans (clean) | woods | Overall of 9 classes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CLODD(contiguous) | Mean | **71.40** | **55.62** | 89.67 | 95.64 | **98.72** | **70.41** | **82.02** | **52.14** | 98.65 | **79.54** |
| CLODD(contiguous) | Weight | 65.30 | 48.73 | 89.67 | 95.48 | **98.72** | 63.57 | 77.86 | 43.38 | 98.65 | 75.59 |
| CLODD(non-contiguous) | Mean | 69.40 | 52.32 | **88.92** | **96.15** | 98.47 | 68.99 | 80.04 | 43.79 | 98.65 | 77.71 |
| CLODD(non-contiguous) | Weight | 65.74 | 47.23 | **88.92** | 95.81 | 98.47 | 64.60 | 78.42 | 43.58 | **98.84** | 75.79 |
| Hierarchical | Mean | 67.31 | 49.03 | 88.16 | 92.63 | 97.70 | 64.21 | 78.37 | 31.16 | 98.45 | 74.94 |
| Hierarchical | Weight | 65.48 | 50.67 | 89.92 | 95.14 | 97.95 | 68.73 | 79.28 | 47.45 | 98.55 | 76.90 |
| $c$-means | Mean | 66.78 | 53.97 | 86.15 | 93.13 | 97.19 | 67.96 | 77.66 | 33.20 | 97.78 | 75.45 |
| $c$-means | Weight | 63.73 | 49.63 | 83.88 | 93.13 | 97.19 | 68.60 | 77.10 | 44.81 | 95.65 | 74.86 |

### 5.2.2 Reordering of DM using iVAT

Figure 5.2(a) shows that some non-contiguous bands are similar. We can group those similar bands together in the matrix if we re-arrange the indicies (bands) using VAT [31]. VAT re-orders bands (data points in standard clustering) based on Prim's modified minimal single linkage. In [88], Havens et al. proposed an *improved VAT* (iVAT) that uses the graph theoretic distance to transform VAT to enhance our visualization and the effectiveness of the VAT algorithm. Figure 5.2(b) is the iVAT enhancement step on (a) without re-ordering and (c) is the enhanced iVAT on the re-ordered DM (b).

### 5.2.3 Clustering of a DM

CLODD, a "visual" clustering algorithm, which is more of an image processing technique than standard feature space clustering, exploits the "blockiness" in the raw DM or a reordered DM. Initially, VAT was created as a tool to help a user 'see" if there is any potential structure in the data. CLODD goes the next step and clusters the data. Its goal is to find a hard partitioning (aka clusters) via dark blocks along the matrix diagonal. While

searching for the partition boundaries, it considers contrast between the on-diagonal dark block and off-diagonal lighter blocks known as "squareness" and visually apparent edges between the blocks, termed as "edginess".

Let $D$ be the DM, $U$ is a $c$ partitioning and $b_i$ is the number of (contiguous) bands in cluster $i$. Squareness is

$$E_{sq}(U;D) = \frac{\sum_{i=1}^{c}\sum_{s\in i,t\notin i} d_{st}}{\sum_{i=1}^{c}(b-b_i)b_i} - \frac{\sum_{i=1}^{c}\sum_{s,t\in i,s\neq t} d_{st}}{\sum_{i=1}^{c}(b_i^2-b_i)}.$$

The first part is the average between dark and non-dark regions. The second is just for dark regions. Edginess is

$$E_{edge}(U;D) = \frac{1}{c-1}\left(\sum_{j=1}^{c-1}\frac{\sum_{i=m_j-1}^{m_j}|d_{i,m_j}-d_{i,m_j+1}|}{b_j+b_{j+1}} + \frac{\sum_{i=m_j+1}^{m_{j+1}}|d_{i,m_j}-d_{i,m_j+1}|}{b_j+b_{j+1}}\right),$$

where $m_j = \sum_{k=1}^{j} b_k$ and $m_0 = 1$.

The objective function has two controlling parameters: mixing coefficient, $\alpha$ to trade-off between squareness and edginess; and $\gamma$ to impose minimum cluster size,

$$E(U,D) = s(\min_{1\leq i\leq c} b_i, \gamma b)(\alpha E_{sq}(U,D) + (1-\alpha)E_{edge}(U,D)),$$

where s(.) is a spline function and is maximized with respect to U to obtain the optimum partition, $U^*$.

### 5.2.4 Feature extraction

Herein, we explore two feature extraction methods, mean and weight. In the 'mean' method, the resultant feature in each band group is the mean value of all bands in that

group. In 'weight', the weight of each band is determined as $W_i = \frac{1}{R} \sum_{j \in C_i, j \neq i} \frac{1}{\epsilon + d(i,j)^2}$. The band with the highest weight, i.e., minimum average distance from all other bands in that group, is selected as the representative [133]. While we could have performed more advanced feature-level fusion or dimensionality reduction methods, the (simple) mean and weight were used as they are more easily translated into a physical sensor when designing a multispectral sensor.

### 5.2.5 Classification

Herein, we use a soft margin *support vector machine* (SVM) with RBF kernel for classification [50]. While we could have used a more sophisticated classifier, e.g., multiple kernel learning [150], we desired to reduce the number of "free parameters" to study just the proposed band grouping technique relative to related work.

## 5.3 Preliminary Findings

The publicly available benchmark data set Indian Pines is used to validate our method. The image has $145 \times 145$ pixels with a spatial resolution of $20$ meters and $220$ spectral channels (bands). We removed $20$ water absorption bands, $104 - 108$, $150 - 163$ and $220$; we consider only those classes with more than 5% of the total samples— Corn-notill, Corn-mint, Grass-pasture, Grass-trees, Hay-windowed, Soybean-notill, Soybean-mintill, Soybean-clean and woods.

We used a random jack-knife partitioning of the data, where 20% are for training and the remainder is testing. We modified CLODD and instead of letting it pick $c$ we varied $c \in \{3, 4, ..., 35\}$ and $\alpha$. We keep $\gamma$ fixed to minimum cluster size of $2$. Whereas CLODD will

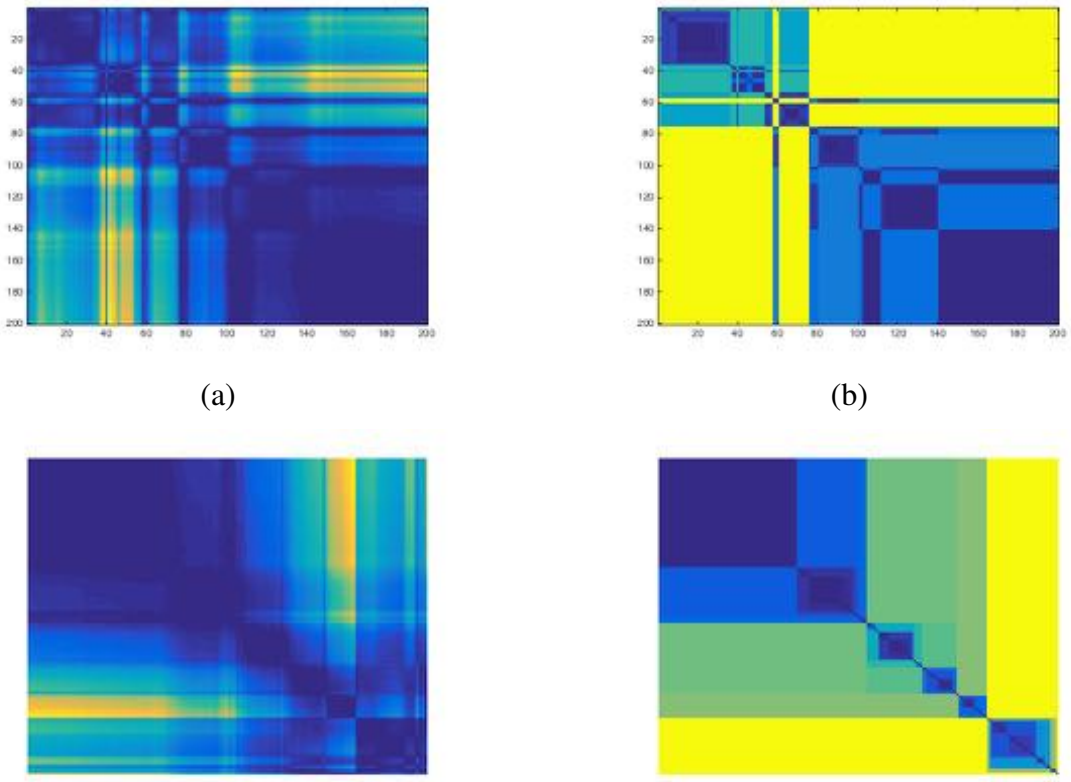
pick the "best visual clustering", we wanted to generate multiple candidate partitions and pick the "winner" based on classification accuracy. Next, the training data is standardized for each feature to have zero mean and unit variance. The testing data is standardized with the mean and standard deviation of the training data. We varied the RBF and adopted a one-vs-all strategy for multi-class classification. The best classification accuracy for each scenario is reported and used for comparison.

Table 5.1 is the classification accuracy for unsupervised band grouping. Non-contiguous CLODD-mean has the best overall performance. For soybean clean, it shows great improvement of approximately 5% over the Hierarchical based method. Note that wood has almost the same accuracy for different methods because its characteristics are distinctly different from all other classes which makes it easily classifiable by any of the methods. On the other hand, corn(min) and corn(notill) have very similar characteristics and are difficult to differentiate. For these two classes, contiguous CLODD-mean is the best.

Supervised band grouping is reported in Table 5.2. Contiguous CLODD-mean is still the top performer and it has a slightly better overall accuracy than unsupervised. For this data set, we see that unsupervised or supervised (how to construct the DM) does not make a significant difference in performance. Another point is performance appears to greatly depends on the band group feature extraction method. For example, CLODD and $c$-means favor the mean whereas Hierarchical likes weight.

## 5.4 Conclusion and Future Work

Herein, we explored a visual clustering algorithm, CLODD, and re-ordering technique, iVAT, for contiguous and non-contiguous band group selection in hyperspectral imaging. Previously, these clustering techniques were used for feature space clustering instead of band group selection. Experimental results indicate that contiguous CLODD is the top performer. However, in future work we will explore the proposed algorithms on additional data sets, compare to more band selection and band group selection algorithms, explore additional DM functions for supervised and unsupervised, feature extraction methods, fusion strategies and more sophisticated classifiers.



(a) (b)

(c) (c)

Figure 5.2
Supervised DM for Indian Pines data set; (a) “raw” DM, (c) VAT re-ordered, (b) iVAT enhanced minus the re-ordering step, and (d) iVAT enhanced with re-ordering.

# CHAPTER VI

# FUSION OF DIVERSE FEATURES AND KERNELS USING LP-NORM BASED MULTIPLE KERNEL LEARNING IN HYPERSPECTRAL IMAGE PROCESSING

## 6.1 Introduction

Hyperspectral imaging has a wide range of applications from mineralogy to weather forecasting, agriculture, surveillance, etc. A hyperspectral image can be described as a high dimensional data cube. Each sub-image in this data cube (usually on the order of hundreds) informs us about the radiance (or reflectance) properties of a scene at different narrowly spaced bands in the *electromagnetic* (EM) spectrum. However, automated analysis of some geographical area might require several remote sensing systems with sensors in different regions of the EM, e.g., visible, near IR and SWIR hyperspectral imagery, lidar and *synthetic aperture radar* (SAR). The point is, numerous sensors are often involved in remote sensing and new theory is needed to fuse them. Herein, we focus on the production and fusion of disparate features in hyperspectral imagery for robust classification. However, without loss of generality, the underlying approach discussed in this article is equally applicable to multi-sensor fusion.

The last decade has seen a surge of interest in the development and use of *multiple kernel learning* (MKL) for tasks like heterogeneous data fusion, classification and input/feature selection in areas like machine learning, pattern recognition, signal processing,

computer vision, etc. A good recent review of MKL mathematics and algorithms is [67]. In the context of *support vector machine* (SVM) based classification, it is often the case that a single kernel is not enough. In practice, a challenge is finding a quality kernel. This is where MK helps. Instead of being restricted to a limited set of known kernels, which may not solve a task, MK provides a solid foundation to combine (fuse) a set of known base kernels (those satisfying Mercer's conditions) to produce a new and more powerful tailored kernel. MK is both a homogenization transformation for different input spaces and it ultimately provides important flexibility for classification in terms of searching for quality linearly separable solutions in the *reproducing kernel Hilbert space* (RKHS). There are number of outstanding MKL problems, for example: how do we generate diverse inputs for MKL; what linear or nonlinear aggregation operators are needed to combine the base kernels; how are multiple kernels normalized; how do we mitigate overfitting in MKL; and what search algorithms are needed for fusion algorithm parameter estimation. To date, numerous algorithms have been put forth, e.g., MKLGL [200], $\ell_p$-norm MKL [125], FIGA [98], GAMKL$_p$ [150], DeFIMKL [150], etc.

In terms of hyperspectral image processing, MKL has been used for tasks like classification [82], feature selection [189] and nonlinear unmixing [39]. In [213], Zhang et al. presents a multi-sensor fusion technique, where $\ell_1$-norm MKL is used to fuse several multi-scale RBF kernels applied to each sensor data set. Majority voting is used to aggregate the MKL classification results. The main contribution was the use of *active learning* (AL) for the selection of training samples based on maximum disagreement. In [189], simpleMKL [4] was used to help learn image features. Multiple RBF kernels are applied

to a single feature, a group of features and features from heterogeneous sources. In [96], Honeine and Richard proved that the angular kernel is a valid reproducing kernel, and it was explored for hyperspectral image processing because spectral angle is a popular tool used in the literature due to its invariance to the spectral energy.

Gu et al. published a series of papers on MKL for classification in hyperspectral imagery [82, 81, 83]. These articles are efficient algorithms to learn the optimum weights in $\ell_1$-norm MKL. The weights of the kernels are obtained by using maximum variance kernel with minimum F-norm error [82], applying low rank *non-negative matrix factorization* (NMF) in [83], and regularizing the weights using cardinality based constraints in [84], which is the extension of [82] for sparse MKL. In [83], *kernel based NMF* (KNMF) uses the non-linear mapping of the base kernels. Specifically it uses the polynomial kernel to map the base RBF kernels to a higher dimensional RKHS. In all the papers, multi-scale RBF kernels based on Euclidean distance is used as the base kernels.

Kloft *et al.* provided an extensive analysis on the different variants of MKL [125]. They showed theoretically and analytically that $\ell_1$-norm MKL has higher performance in noisy situations where the noisy kernels are eliminated via the sparse weights. On the other hand, higher norm MKL tends to give equal weights to all the kernels, and therefore outperforms the sparse MKL when the kernels are good and diverse. In many cases in hyperpsectral image processing, we can have diverse feature sets that can be used to generate quality kernels. This in turn signifies that dense MKL has a huge potential to improve the classification results over the sparse MKL, but it has not been explored in the hyperspectral

community to date. Herein, we employed the $\ell_p$-norm MKL and found similar results that support the analysis in [125].

In summary, while there has been interest in using MKL for hyperspectral image processing, work to date has primarily focused on the fusion of homogeneous kernels, e.g., multiple RBFs with Euclidean distance. However, it is likely that different kernels are required. In addition, to the best of our knowledge little-to-no work has focused on how to generate a diverse set of features for MKL via bandgrouping in hyperspectral image processing. We show that diversity with respect to both features and kernels is important for MKL as well as $\ell_p$-norm MKL outperforms sparse MKL in aggregating them. Figure 6.1 is a high-level illustration of our approach.

## 6.2 Methods

In this section, we describe the three major parts of our approach—(i) proximity measure calculation, (ii) band grouping for feature extraction, and (iii) feature space fusion using $\ell_p$-norm MKL. For notational purposes, the 3D hyperspectral data cube is remapped into a 2D space such that each row represents a pixel and each column is a band. Let the re-shaped 2D data set be $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n]^T \in \mathbb{R}^{n \times b}$, where $n$ is the number of pixels in the image and $b$, the number of bands.

### 6.2.1 Proximity Measure Calculation

Hyperspectral sensors are wonderful because each pixel has a wealth of information and tells a story, versus traditional single channel or "RGB" imagery. However, hyperspectral imagery also suffers from the curse of dimensionality, spatially, spectral and sometimes

temporal. Instead of using all bands, or features, it is often the case that selecting individual bands, feature projection (e.g., *Principle Component Analysis* (PCA), random projection, etc.), or grouping bands leads to a better solution (higher accuracy and more robust). While numerous approaches have been proposed, it has been shown that band group partitioning is of utility and it can be derived, in a supervised or unsupervised fashion, from a proximity measure [22]. For example, one can compute the correlation matrix between the different bands (unsupervised approach). Structure in this matrix can be used to derive a band group partitioning. However, there are a number of unsupervised proximity measures, such as Euclidean, correlation, Jeffrey K. Matusita, Bhattacharyya, *spectral angle mapper* (SAM), etc. In general, it has been demonstrated that selection of proximity measure depends in part on the data set and task, meaning there does not appear to be a global best. Herein, we explore the square of Euclidean, which measures the distance between a pair of vectors, and correlation, which measures angular similarity. We selected these two proximity measures for demonstration as they capture different aspect of the data via its features. However, in future work this is likely a parameter that needs to be included in our algorithm.

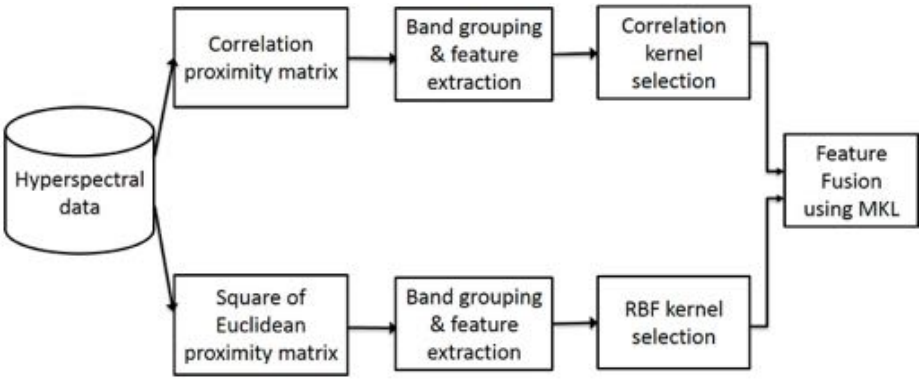



Figure 6.1
High-level illustration of the proposed MKL approach.

**Proximity Measure 1: Square of Euclidean** The square of Euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is

$$d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{L} (x_{ik} - x_{jk})^2.$$

Note, the square of Euclidean distance is always positive and depends in large on $L$, e.g., the length of the vectors or the number of pixels in the training data set.

**Proximity Measure 2: Correlation** Correlation is a similarity measure between two signatures. The Pearson's correlation coefficient is the co-variance of two vectors normalized by the product of the standard deviation of two distributions,

$$s(\mathbf{x}_i, \mathbf{x}_j) = corr(\mathbf{x}_i, \mathbf{x}_j) = \frac{cov(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_j}}.$$

The correlation coefficient is in $[-1, 1]$. Distance, or the dissimilarity measure, $d(\mathbf{x}_i, \mathbf{x}_j)$ is obtained herein by simply subtracting the $s(\mathbf{x}_i, \mathbf{x}_j)$ from 1.

### 6.2.2 Feature Extraction

In this step, we first partition similar bands from a given proximity measure into groups and we then apply a feature extraction or reduction technique to each band group to extract a single feature from that group. Herein, we use the algorithm proposed by Ball *et al.* that performs unsupervised grouping of contiguous similar bands with respect to a provided proximity measure (see [22] for full algorithm details). After band grouping, we can apply a number of feature extraction techniques such as *stepwise linear discriminant analysis* (SLDA) [23], mean or weight, to get features equal to the number of band groups. In this paper, we calculate the mean of all bands in a groups as the feature. While mean might not

be the most sophisticated technique, its advantage is that it is simple to realize in hardware and gives rise to a simpler multispectral versus hyperspectral sensor.

### 6.2.3 Feature Space Fusion Using $\ell_p$-Norm MKL

In the kernel approach, inputs are ideally projected into a high, possibly infinite, dimensional RHKS space, where the patterns for different classes are now linearly separable. The trick is that we can do this all via a "kernel function" in the original low(er) dimensional space and we never have to do the actual lifting. However, in reality we do not know what kernel to select and in general the choice of kernel is task specific. There is currently no straightforward way to select a kernel for a given set of data. As already mentioned, MKL provides one such path to help search for the idea kernel by the simple concept of combining simple known (base) kernels to form custom (tailored) kernels. However, we must search for this kernel and the space is both extremely large and if we are not careful, MKL tends to succumb to overfitting (can learn the training data perfectly but not generalize well to new test data). Herein, we restrict our analysis to a *linear convex sum* (LCS) of kernels. This is by far the predominant MKL approach. While a few nonlinear approaches have been put forth, e.g., FIGA, for various reasons (such as proving that a given aggregation operator always yields a valid Mercer kernel) nonlinear MKL is still an unsolved problem.

For a function to be a kernel, it must satisfy the Mercer's kernel properties such as continuity, symmetry, and positive semi-definiteness. There are numerous kernel func-

tions, e.g., *radial basis function* (RBF), polynomial, etc. In this paper, we use RBF and correlation kernel. The RBF function is

$$k_r(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{||\mathbf{x}_i^2 - \mathbf{x}_j^2||}{2\sigma^2}\right),$$

where $\sigma$ is the so-called width parameter of the RBF kernel. The correlation kernel is

$$k_c(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{1 - corr(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right)$$

where $corr(\mathbf{x}_i, \mathbf{x}_j)$ is the Pearson's correlation coefficient for $\mathbf{x}_i$ and $\mathbf{x}_j$. In [113], the authors have shown that the correlation kernel satisfies the Mercer's kernel properties. Note, our two kernels are already more-or-less to scale by design. However, if one is using heterogeneous kernels that produce very different scales, then the zero mean and unit variance RHKS approach in [125] can be used.

The convex sum of $M$ kernels is also a Mercer's kernel. This is because both the sum and multiplication by positive constant are *positive semidefinite* (PSD) preserving operators (on $M$ different Gram matrices). The combined kernel with $\ell_p$-regularized weight $w_m$ is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{M} w_m k_m(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $||\mathbf{w}||_p \leq 1$ and $w_m \in \mathbb{R}^+$, where $||\mathbf{w}||_p$ is the $p$-norm of $\mathbf{w}$. Though the above expression is notationally for $M$ kernels on the same set of features, it is trivially generalized to multiple features, e.g., different kernels on different subsets of features [150]. Optimization-based MKL solutions, versus fixed rule or heuristic approaches, optimize (using alternating optimization typically) the weights of the kernels and the SVM criteria function. Again, we use $\ell_p$-norm MKL [82, 189] to derive the LCS weights. However, we

could use a number of other search algorithms for feature level fusion, such as MKLGL$_p$ or GAMKL$_p$, or decision-level MKL, e.g., DeFIMKL$_p$. The $\ell_p$-norm condition is more-or-less the same across a number of solvers. In general, the different approaches represent variations in search, e.g., Group Lasso (MKLGL), genetic algorithm based (GAMKLp), and non-linear decision-level fusion via DeFIMKL.

### 6.3 Preliminary Results and Discussion

The Indian Pines data set consists of $145 \times 145$ pixels with a spatial resolution of $20$ meters and $220$ spectral channels (bands). During data pre-processing of the data, $20$ water absorption bands, $104 - 108$, $150 - 163$ and $220$ were removed. We considered the following $9$ classes for classification — Corn-notill, Corn-min, Grass-pasture, Grass-trees, Hay-windowed, Soybean-notill, Soybean-mintill, Soybean-clean and woods. Random jack-knife partitioning is used, where 20% is training and the remainder is testing. Hereafter, the squared Euclidean is denoted as 'SqE' and correlation is 'Corr'. Proximity matrices are computed on the training data based on SqE and Corr. The number of band groups and thus features extracted was $11$ for SqE and $16$ for Corr (forming two feature vectors). The training data is standardized for each feature to have zero mean and unit variance and the testing data is standardized using the mean and standard deviation of the training data. For SqE and Corr, we used 10 kernels each with $\sigma = \{2^{-3}, 2^{-2}, \cdots, 2^{6}\}$. Top performing kernels for each feature were selected using SVM accuracy and fused using $\ell_p$-norm MKL. 'SVMLight' and 'MKL' implementations in the Shogun toolbox [171]

were used. For the $\ell_p$-norm, we tried $p = 1.1$ (approximately city block distance), $p = 2$ (Euclidean) and $p = 100$.

Table 6.1
Producer's accuracies for $\ell_p$-norm MKL based fusion.

| $\ell_p$-norm | Method (SqE = Square of Euclidean, Corr = Correlation) | # of kernels | corn (notill) | corn (min) | grass (pasture) | grass (trees) | hay (windrowed) | soybeans (notill) | soybeans (min) | soybeans (clean) | woods |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NA | SqE | 1 | 64.52 | 46.18 | 75.57 | 92.80 | 97.19 | 63.82 | 78.57 | 28.31 | 97.78 |
| | Corr | 1 | 62.42 | 34.93 | 81.11 | 85.09 | 96.16 | 62.02 | 66.46 | 30.55 | 97.49 |
| $p = 1.1$ | Fusion of SqE & Corr | $1 + 1$ | 68.35 | 46.48 | 79.60 | 90.79 | 96.16 | 64.99 | 79.08 | 45.42 | **97.87** |
| | SqE | 2 | 65.91 | 52.62 | 87.41 | 92.29 | 97.44 | 64.34 | 80.09 | 37.27 | 97.78 |
| | Corr | 2 | 62.77 | 36.13 | 82.12 | 87.27 | 96.16 | 62.02 | 66.41 | 30.55 | 97.49 |
| | Fusion of SqE & Corr | $2 + 2$ | 68.70 | 52.02 | 88.41 | 93.13 | 96.42 | 64.99 | 80.45 | 46.64 | 97.87 |
| $p = 2$ | Fusion of SqE & Corr | $1 + 1$ | 69.92 | 53.37 | 82.87 | 91.96 | 96.16 | 69.77 | 80.14 | 52.55 | 97.20 |
| | SqE | 2 | 69.14 | 57.12 | 88.66 | 92.80 | 97.70 | 69.51 | 81.81 | 46.64 | 97.78 |
| | Corr | 2 | 65.82 | 39.28 | 87.15 | 88.11 | 97.19 | 63.70 | 66.67 | 37.07 | 97.49 |
| | Fusion of SqE & Corr | $2 + 2$ | 73.50 | 62.82 | 91.18 | 93.97 | 97.70 | 72.22 | 83.23 | 60.08 | 97.10 |
| $p = 100$ | Fusion of SqE & Corr | $1 + 1$ | 71.49 | 60.57 | 83.63 | 93.63 | 96.93 | 72.87 | 81.16 | 60.29 | 96.14 |
| | SqE | 2 | 72.97 | 64.32 | 89.67 | 94.47 | 97.70 | 73.00 | 83.13 | 55.80 | 97.29 |
| | Corr | 2 | 68.88 | 46.03 | 89.92 | 89.61 | 97.44 | 66.93 | 67.68 | 45.42 | 97.39 |
| | Fusion of SqE & Corr | $2 + 2$ | **77.24** | **69.42** | **92.44** | **94.97** | **97.95** | **76.49** | **85.11** | **66.40** | 95.75 |

Table 6.1 shows that inter-method fusion, i.e., fusion of SqE and Corr, is the top performer for all classes. Also, a larger $p$ produces the best results for all classes except 'woods'. Inter-method fusion has an improvement of approximately $2\%$ to $10\%$ relative to intra-method for corn-notill, corn-min, grass-pasture, soybeans-notill, soybeans-min and soybeans-clean. It has almost the same performance for grass-tress and hay-windowed. The behavior of wood is different from all other classes. It has the best result at $p = 1.1$, and it continues to degrade with increasing $p$. Note, 'wood' is easily classifiable with a single kernel. $\ell_{1.1}$ MKL, which promotes sparse solutions, is more suitable for this task. In [125], Kloft et al. showed that when kernels are diverse, higher norm MKL is more appropriate and yields better results. In our case, results improve for $8$ out of $9$ classes as

$\ell_p$-norm increases, which supports our claim that diversity in features and kernels is useful for MKL-based hyperspectral classification.

## 6.4 Future Work

As stated above, our results are preliminary but promising—ran on a single well-known benchmark data set that was not trivial to solve using a single kernel. In future work, we will apply our technique to additional data sets. We will also investigate a search procedure for MKL parameter selection, including kernel type and associated parameters (a critical aspect of MKL that is typically overlooked due to complexity). Here, we explored, for demonstration, one kernel based on Euclidean distance and another based on angle for diversity. We believe it is also of interest to explore different, or a combination of different band group selection algorithms and what particular proximity measures are fed to these techniques to ultimately generate diverse features for MKL. Last, we are currently using all features produced by band grouping. However, sometimes some bands (or band groups) are not useful for a task at hand and performance can be raised if we do a feature selection step before fusion.

# CHAPTER VII

# MULTI-CRITERIA BASED LEARNING OF THE CHOQUET INTEGRAL USING GOAL PROGRAMMING

## 7.1 Introduction

The *fuzzy integral* (FI) has been used time and time again in numerous applications such as signal and image processing, pattern recognition and *multi-criteria decision making* (MCDM) for data/information aggregation. In 1974, Sugeno introduced the *fuzzy measure* (FM), a monotone and normal capacity [176]. The FM is significant with respect to the FI because it is in effect what drives (determines what specific function is computed) nonlinear aggregation via FIs like the *Choquet integral* (CI) and *Sugeno integral* (SI). Existing approaches either manually specify the FM or attempt to learn it from data according to a criteria such as the *sum of squared error* (SSE). However, the FM is difficult to specify as it has $(2^N - 2)$, for $N$ inputs, numbers of "free parameters". It is extremely rare than an expert can (or would want to) provide such information even for relatively small cases, e.g., $N = 4$ has $14$ values, $N = 5$ has $30$ and $N = 10$ already has $1,022$ values. Common practice is to specify just the *densities*, the measure values for only the singletons (individuals). From there, we can use methods such as the S-Decomposable FM, the Sugeno FM [176], and Grabischs k-additive FM and integral [140]. In [79], Grabisch and Roubens proposed a method known as constraint satisfaction that takes the relative importance of the

criteria and the type of interaction between them, if any. Alternately, identification of FMs based on data has been explored by numerous authors in different applications; *quadratic programming* (QP) [76], gradient descent [121], penalty and reward [120], Gibbs sampler [136], and genetic algorithms [11]. However, to the best of our knowledge no method has been put forth to date to learn the FM by taking into account a weighted combination of both experts' knowledge and data.

One problem with learning an aggregation from data alone is that it often results in solutions that are overly complex and "expensive" to implement. It also runs the risk of over-fitting and the quality of that solution is based in large on the size and diversity of the data. On the other hand, learning an aggregation based on only expert opinion can be overly subjective and may not result in peak performance. However, conflict between data sets, experts or a combination of the two, can (and does) occur and must be addressed.

Consider the following multi-sensor target detection example. The goal is to use the individual (and most important, combined) benefit of different sensors operating across the electromagnetic spectrum. While we want to achieve the highest performance possible, we also desire a solution whose overall cost does not exceed some limit, one that requires the fewest computational resources (e.g., memory and processing), has the smallest form factor and energy consumption, etc. At the heart of such a dilemma is the need to optimize some process relative to different and often conflicting information.

Herein, weighted Goal programming [100], an approach from MCDM, is explored to learn the FM relative to the CI for data/information fusion. This framework provides an interesting new way to set the priority of any number of combination of different sources

(e.g., data and high-level expert knowledge) in learning. Specifically, relative weights are placed on data and experts. The user can exercise control on how and to what extent the criteria from each input contribute to learning. Figure 7.1 shows the proposed weighted Goal programming framework for CI learning.

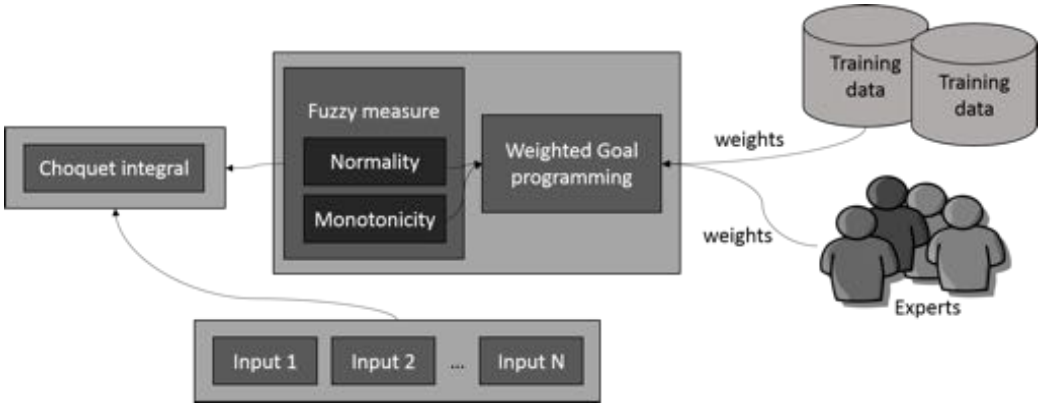



Figure 7.1
Proposed weighted Goal programming framework to learn the CI from a combination of data and high-level expert knowledge. Relative priorities (weights) are placed on training data and experts. Weighted Goal programming also allows for a way to enforce the hard constraints imposed by the fuzzy measure (normality and monotonicity).

This paper is organized as follows. In Section II, we review important definitions and we discuss capacity learning based on training data by SSE minimization. In Section III, after brief introduction on weighted goal programming, our method of aggregation based on training data and experts criteria is explained in detail. Last, in Section IV, experiments for different scenarios are provided and analyzed.

## 7.2 Background

We begin with a few necessary definitions relevant to data/information aggregation. This is followed by a description of FM learning using the QP. While several FIs exist,

e.g., the SI and CI, the focus of this article is just the CI. The CI has numerous desirable properties, i.e., it is differentiable, it recovers the Lebesgue integral for an additive FM, etc. The CI is defined with respect to the FM, which ultimately determines what specific aggregation operator the CI breaks down into. Let there be $N$ inputs, $X = \{x_1, x_2, \cdots, x_N\}$ and a (partial support) function, $h$, that maps each element of $X$ into an interval such as $[0, 1]$. For example, $X$ can be set of algorithms, features, sensors, etc. Furthermore, the integrand ($h$) are often labels in pattern recognition, confidences, utility in decision making process, evidence or sensor (quantitative) measurements. Note, the $[0, 1]$ interval convention is a matter of convince. Other intervals can (and have) been explored in the literature. The FM, for $\forall A \subseteq X$, is a value in $[0, 1]$ that often represents the (relative) "worth" of the different subsets of sources. Often these values are subjective (e.g., provided by a human), however they can be objective as well (e.g., correlation in a multi-sensor system). The CI is a unique and creative way of combining the information in both g and h.

**Definition 1. (Fuzzy Measure)** The fuzzy measure is a set valued function
$g : 2^X \rightarrow [0, 1]$ such that

1. (Boundary condition) $g(\emptyset) = 0$;
2. (Monotonicty constraints) If $A, B \in X,$ and $A \subseteq B$, then $g(A) \leq g(B)$.

Note, often $g(X) = 1$ (normality) is enforced as an additional boundary constraint. Also, if $X$ is an infinite set then a third condition guaranteeing continuity is required, but this is

a moot point for finite $X$. For example, for three inputs, we have the following boundary constraints

$$g(\phi) = 0,$$

$$g_{123} = 1,$$

and the monotonicity constraints are

$$g_1 - g_{12} \leq 0,$$

$$\cdots,$$

$$g_{23} - g_{123} \leq 0,$$

where $g_{1,\cdots,i}$ is a lexicographically encoded representation (short hand notation), i.e., $g_1 = g(x_1), g_{13} = g(\{x_1, x_3\})$ etc.

**Definition 2. (Training Data)** Let the training data set be

$$T = \{(O_j, \alpha_j) : j = 1, \cdots, m\},$$

where $\mathbf{O} = \{O_1, \cdots, O_m\}$ is the set of objects and $\alpha = \{\alpha_1, \cdots, \alpha_m\}$ are the corresponding labels. For example, in signal/image processing and computer vision an object could be an image chip and $O_i$ ($1 \leq i \leq N$) could be the output (e.g., class label) of an algorithm asserting the support that a human is in the image chip. Next, we provide the definition of the CI with respect to training data.

**Definition 3. (Choquet Integral)** The discrete CI for $O_j$ is

$$C_g(h(O_j)) = \sum_{i=1}^{N} [h(O_j : x_{\pi(i)}) - h(O_j : x_{\pi(i+1)})] g(A_{\pi(i)}),$$

for $A_{\pi(i)} = \left\{x_{\pi(1)}, \cdots, x_{\pi(i)}\right\}$ and permutation $\pi$ such that

$$h(O_j : x_{\pi(1)}) \geq \cdots \geq h(O_j : x_{\pi(N)}),$$

where $h(O_j : x_{\pi(N+1)}) = 0$.

Next, we proceed to the formulation of the SSE minimization problem for FM learning and the QP. Let the SSE between training data $T$ and the CI with respect to the FM $g$, be [12]

$$E(g) = \sum_{j=1}^{m} (C_g(h(O_j)) - \alpha_j)^2,$$

which can be expanded as

$$E(\mathbf{u}) = \sum_{j=1}^{m} (A_{O_j}^t \mathbf{u} - \alpha_j)^2, \tag{7.1}$$

where

$$\mathbf{A}_{O_j} = \begin{pmatrix} \cdots \\ h(O_j : x_{\pi(1)}) - h(O_j : x_{\pi(2)}) \\ \cdots \\ h(O_j : x_{\pi(i)}) - h(O_j : x_{\pi(i+1)}) \\ \cdots \\ 0 \\ \cdots \\ h(O_j : x_{\pi(N)} - 0) \\ \cdots \end{pmatrix},$$

and

$$\mathbf{u} = [g_1, g_2, \cdots, g_{12}, \cdots, g_{12\cdots N}]^t.$$

Both $\mathbf{A}_{O_j}$ and $\mathbf{u}$ are of size $(2^N - 1) \times 1$. The function difference $h(O_j : x_{\pi(i)}) - h(O_j : x_{\pi(i+1)})$ corresponds to location $g(x_{\pi(1)}, \cdots, x_{\pi(N)})$ in $\mathbf{u}$. Equation (7.1) can be expanded and simplified to give the following form

$$E(\mathbf{u}) = \mathbf{u}^t \mathbf{D} \mathbf{u} + \mathbf{f}^t \mathbf{u} + \sum_{j=1}^{m} \alpha_j^2,$$

where

$$\mathbf{D} = \sum_{j=1}^{m} \mathbf{A}_{O_j} \mathbf{A}_{O_j}^t, \; f = \sum_{j=1}^{m} (-2\alpha_j \mathbf{A}_{O_j}).$$

The CI is learned through minimizing an objective function such as Equation (7.1) with respect to the monotonicity and boundary conditions of the FM. For $N$ inputs, the FM has $2^N$ variables, of which two variables $g(\emptyset)$ and $g(X)$ have constant values i.e., $g(\emptyset) = 0$, and $g(X) = 1$. Therefore, the monotonicity constraints can be constructed with the remaining $2^N - 2$ variables without considering these two constant valued variables, however, we have represented it with $2^N - 1$ variables including $g(X)$ in matrix form (of size $N(2^{N-1} - 1) \times (2^N - 1)$) by [12],

$$\mathbf{C}_M \mathbf{u} \leq \mathbf{0}$$

where,

$$\mathbf{C}_M = \begin{pmatrix} \\ \boldsymbol{\Psi}_1^t \\ \\ \boldsymbol{\Psi}_2^t \\ \\ \cdots \\ \\ \boldsymbol{\Psi}_{N+1}^t \\ \\ \cdots \\ \\ \boldsymbol{\Psi}_{N(2^{N-1}-1)}^t \end{pmatrix}.$$

Here, $\boldsymbol{\Psi}_1$ is the vector representing the coefficients of the monotonicity constraint 1, $g(1) - g(12) \leq 0$. So, the coefficient matrix of monotonicity constraints consists of $\{0, 1, -1\}$ [12]

$$\mathbf{C_M} = \begin{pmatrix} 1 & 0 & \cdots & -1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

Finally, we can write the SSE minimization problem in the QP form as

$$\min_u \frac{1}{2}\mathbf{u}^t\widehat{\mathbf{D}}\mathbf{u} + \mathbf{f}^t\mathbf{u}$$

subject to

$$\mathbf{C}_M\mathbf{u} \geq \mathbf{0}$$

$$(\mathbf{0}, 1)^t \leq \mathbf{u} \leq \mathbf{1}.$$

Note, $\widehat{D} = 2D$ and our inequality need only be multiplied by $(-1)$ in order to formulate the underlying QP.

## 7.3 Methods

In this section, the techniques to solve the FM learning problem relative to the CI based on both expert opinions as well as training data is discussed. This problem could be solved relatively easily by augmenting the constraints of the training data only problem with the expert constraints in the event that the newly added constraints do not bring about any conflict. Ascertaining any violation by only observing the expert criteria is difficult, and would be of no help because real world problems can (and often do) involve conflicting criteria. Therefore, we explore a method that employs weighted Goal Programming to address conflicting criteria. For example, conflict among experts and expert versus data. The proposed method involves building criteria with respect to training data by transforming the SSE objective and also mapping the expert opinion criteria into a set of inequality and equality relations. In the subsequent subsections, we give a brief description of weighted Goal Programming and discuss monotonicity constraints for the FM relative to CI, FM learning relative to the CI and training data and the case of multiple experts.

### 7.3.1 Introduction to weighted Goal programming

Goal programming is a technique from MCDM for deriving satisfying solutions from conflicting multi-objective Goals subject to a set of constraints. As for conflicting Goals, simultaneous achievement of all Goals are not possible. Therefore, Goal programming tries to attain each goal as close as possible. This is accomplished by introducing deviation variables in each goal and defining an objective to minimize a function formed with

deviation variables or deviation control parameters. Interested readers seeking in-depth knowledge and examples on Goal programming may explore [101].

Goal programming has several variants like Lexicographic Goal programming, Weighted Goal programming and Chebysheve Goal programming [101]. In Lexicographic Goal programming, the goals, categorized into priority order, are optimized in a way that the goals achieved for the higher priority level must not be worsened while optimizing the next lower priority goals. In Weighted Goal programming, the decision maker expresses preference on goals by relative weights and the undesired deviation of each goal is minimized according to these weights. Additional details can be found at [101]. Weighted Goal programming has been chosen over other variants because it gives the flexibility to set priority quantitatively to any proportion of choice.

### 7.3.2 Minimization of the SSE

As discussed in the previous section, the CI with respect to training data can be learned by minimizing an objective function that corresponds to the SSE between the CI and the label of the data. Using the SSE function defined in Equation (7.1), the minimization problem can be written as

$$\min_{\mathbf{u}} \sum_{j=1}^{m} \left( \mathbf{A}_{O_j}^{t} \mathbf{u} - \alpha_j \right)^2 \tag{7.2}$$

subject to $\mathbf{C}_M \mathbf{u} \geq \mathbf{0}$ and our boundary conditions.

As the Goal programming works on constraints with targets to be attained the proximity as nearest as possible, we have to transform the SSE minimization objective in (7.2) to a set of Goal constraints. The conditions for the minimum value of this convex function

is obtained by computing the gradient of this function along all the hyper-axes and then equating the gradients to zero. We start by expanding Equation (7.2),

$$E(\mathbf{u}) = \sum_{j=1}^{m} (\mathbf{u}^t \mathbf{A}_{O_j} \mathbf{A}_{O_j}^t \mathbf{u} - 2\alpha_j \mathbf{A}_{O_j}^t \mathbf{u} + \alpha_j^2),$$

we then take its derivative,

$$\sum_{j=1}^{m} (2\mathbf{A}_{O_j} \mathbf{A}_{O_j}^t \mathbf{u} - 2\alpha_j \mathbf{A}_{O_j}) = \mathbf{0},$$

and the equality Goal constraints are therefore

$$\sum_{j=1}^{m} \mathbf{A}_{O_j} \mathbf{A}_{O_j}^t \mathbf{u} = \sum_{j=1}^{m} \mathbf{A}_{O_j} \alpha_j, \tag{7.3}$$

or

$$\mathbf{C}_D \mathbf{u} = \mathbf{b}_D,$$

$$\mathbf{C}_D = \sum_{j=1}^{m} \mathbf{A}_{O_j} \mathbf{A}_{O_j}^t, \mathbf{b}_D = \sum_{j=1}^{m} \mathbf{A}_{O_j} \alpha_j,$$

which represents the target. In addition to the monotonicity constraints, constraints representing expert opinions can also be be enforced. This is described next.

### 7.3.3 Expert Opinion

Expert knowledge, when available, is an extremely valuable resource. The goal of this work is to include it in aggregation or aggregation operator learning. However, we must consider inter and intra observer error across experts. In addition, it is a possibility that human input might violate the monotonicity constraints of the FM and/or be in disagreement with training data. Herein, we take into account all of these potential conflicts and aberrations and we perform a weighted combination of hard and soft constraints.

Often, experts provide their opinions as partial order relations of the different sources, e.g., source 1 is more important than source 2. In terms of the FM, we can express this via $g(x_1) \geq g(x_2)$, or alternatively (in linear algebra form) as

$$\mathbf{\Phi}_i^t \mathbf{u} \geq 0,$$

where $i$ indicates the ith input (partial order relation) from our expert, $\mathbf{\Phi}_i$ is of length $(2^N - 2) \times 1$ and this vector has values in $\{1, -1, 0\}$. For the inequality $g(x_1) \geq g(x_2)$, $\mathbf{\Phi}_i$ is

$$\begin{bmatrix} 1 & -1 & \cdots & 0 \end{bmatrix}^t.$$

Suppose, $l$-th expert provides S inequality criteria or constraints. We can express all those constraints as

$$\mathbf{C}_{E_l}^{IN} \mathbf{u} \geq \mathbf{0}, \tag{7.4}$$

where

$$\mathbf{C}_{E_l}^{IN} = \begin{pmatrix} \mathbf{\Phi}_1^t \\ \mathbf{\Phi}_2^t \\ \cdots \\ \mathbf{\Phi}_{S_l}^t \end{pmatrix},$$

where $S_l$ is the number of inequality constraints or criteria from $l$-th expert.

Typically, it is far easier, and more realistic, for an expert to qualitatively express knowledge between different subsets of sources in terms of inequalities versus assigning specific values to the different subsets (aka equality constraints). Nevertheless, our mathematical formulation can accommodate for equality constraints if present. If an expert spec-

ifies that the "worth" of source 3 is 0.1, then we can simply express this as $g(x_3) = 0.1$. This knowledge can be represented in vector form as

$$\Upsilon_i^t \mathbf{u} = b,$$

where

$$\Upsilon_i^t = [0 \quad 0 \quad 1 \quad \cdots 0],$$

$$b = 0.1.$$

Here, a 1 appears in the 3rd index in $\Upsilon_i^t$ which corresponds to $g_3$ in $\mathbf{u}$. If we have equality inputs from $l$-th expert, then we can pack this into matrix form as

$$\mathbf{C}_{E_l}^{EQ} \mathbf{u} = \mathbf{b} \tag{7.5}$$

where

$$\mathbf{C}_{E_l}^{EQ} = \begin{pmatrix} \Upsilon_1^t \\ \Upsilon_2^t \\ \cdots \\ \Upsilon_{Q_l}^t \end{pmatrix},$$

where $\mathbf{b}$ is a column vector of length $Q_l$ and it includes the capacities of the inputs as specified by the expert. Next, we describe how expert knowledge can be utilized along with training data based on Goal programming.

### 7.3.4 Goal programming for training data and multiple experts

In this subsection, the Goal programming problem is formulated for the case of one or more training data sets and multiple expert's opinion with a priority order represented by

relative weights. In standard weighted Goal programming, the objective function consists of single goal minimization of the weighted sum of the undesired deviation. In [104], the author proposed an improved weighted Goal programming method that takes relative normalized weight for the priority of the criteria as well as maintains the relative priority of each goal. The objective function now consists of two goals. The first goal, i.e., term corresponds to the maximum unwanted deviation, and the second term corresponds to sum of the undesired deviations. This method involves lexicographical minimization of the two terms, i.e., minimization of the first term with highest priority and then minimization of the second term without deteriorating the first goal. Let the normalized weight on the training data be $w_D$ and the weights on multiple ($P$) experts be $w_E = \{w_{E_1}, w_{E_2}, \cdots, w_{E_P}\}$ with the normality constraint $w_D + \sum_{l=1}^{P} w_{E_l} = 1$. The weighted Goal program [104] is as follows,

$$\text{Lex}\min z = \left\{ \beta, \sum_i (n_i + p_i) + \sum_l \left( \sum_j n_{lj} + \sum_k (n_{lk} + p_{lk}) \right) \right\}, \tag{7.6a}$$

subject to

$$\mathbf{c}_{D_i}\mathbf{u} + n_i - p_i = b_{D_i}, i = 1, 2, ..., 2^N - 2, \tag{7.6b}$$

$$\mathbf{c}^{IN}_{E_{l_j}}\mathbf{u} + n_{lj} \geq b^{IN}_{E_{l_j}}, l = 1, 2, ..., P, \tag{7.6c}$$

$$\mathbf{c}^{EQ}_{E_{l_k}}\mathbf{u} + n_{lk} - p_{lk} = b^{EQ}_{E_{l_k}}, l = 1, 2, ..., P, \tag{7.6d}$$

$$\mathbf{C}_M\mathbf{u} \geq \mathbf{0}, \tag{7.6e}$$

$$w_D(n_i + p_i) \leq \beta, \tag{7.6f}$$

$$w_{E_l} n_{lj} \leq \beta, \tag{7.6g}$$

$$w_{E_l}(n_{lk} + p_{lk}) \leq \beta, \tag{7.6h}$$

$$n_i, p_i, n_{lj}, n_{lk}, p_{lk} \geq 0, \tag{7.6i}$$

$$(\mathbf{0}, 1)^t \leq \mathbf{u} \leq \mathbf{1}, \tag{7.6j}$$

where $\mathbf{c}_{D_i}$ and $b_{D_i}$ corresponds to the i-th row of the coefficient matrix $\mathbf{C}_D$ and target vector $\mathbf{b}_D$ of the training data constraints. Similar definition applies to $\mathbf{c}_{E^{IN}_{l_m}}$, which is the m-th row of the coefficient matrix of l-th expert's inequality constraints. The parameter $\beta$ minimizes the maximum undesired deviation while maintaining the priority order of the constraints. Equations (7.6e) to (7.6j) are the hard constraints which are always satisfied. The method in [101] applies normalization on each criteria or constraints with respect to target. However, we have not normalize the target values of criteria (e.g., $b_{D_i}$) since the target values are either zero or close to zero.

Above, we restricted the mathematical formulation to a single data set and multiple experts. However, we can utilize the underlying concept of the weighted Goal programming to generalize the framework to cover any number of data sets and multiple experts. For this

more general case, there are (at least) two strategies for assigning relative weights. In the first approach, the priority order or relative weight assignment can be similar to that of a single data set and multiple experts. For example, if we have two data sets and two experts, then the relative weights could be 0.2, 0.3, 0.1, and 0.4 on data set 1, data set 2, expert 1 and expert 2 respectively. So, in terms of weights, it does not differentiate between data sets and experts.

In the second approach we prioritize the group of data sets and group of experts first. At the next level, each member data set is prioritized among the group of data sets. The same principle is followed for experts. For example, consider the case of two data sets and two experts. In the first level, the weight on the group of data sets might be 0.4 while the experts is 0.6. Next, we assign weights to each data set, i.e., data set 1 with weight 0.3 and data set 2 with weight 0.7. Subsequently, the weights of experts could be 0.7 and 0.3. This strategy is referred to as multi-level weighted goal programming.

Next, we generate the equality constraints for all the training data sets as given by (7.3), inequality and equality constraints for each expert according to (7.4) and (7.5). With all the constraints in hand, it is simple to formulate the weighted Goal programming problem in which the objective is to achieve the goal of minimizing the deviation from the goal of each criteria according to their relative priorities. Depending on the weight assignment strategy, the objective function will optimize for all data sets and experts in a single level weight case or we sequentially minimize the deviation of goal for groups (e.g., group of data sets) then for each criteria (e.g., each data set).

**Example 5.** Consider the following challenge from multi-sensor signal processing. Assume high-level expert knowledge is present and three sensors are available for some target detection task. Specifically, assume three algorithms exist, one for each sensor. Without loss of generality, this example is easily extended to the case of a single (or multiple) sensor(s) with different algorithms or the same algorithm with different parameters or trained on different data. Furthermore, assume that a user assigned an importance of 0.8 to the training data, in order to learn a solution that primarily minimizes function error, expert 1 (E1) was assigned a weight ($w_{E1}$) of 0.1 and expert 2 (E2) was assigned a weight ($w_{E2}$) of 0.1. Let expert 1 have information regarding the "complexity" of the various algorithms, e.g., computationally and/or memory utilization wise. This information is important as it dictates the processing hardware (and therefore the financial cost of the system) and it determines the system's response time. E1 provides this information as partial order relations about the relative worth of the algorithms; e.g., $g_4 \succeq g_1$ ,and $g_{12} \succeq g_3$. The constraints constructed from E1's criteria are

$$g_4 - g_1 \geq 0,$$

$$g_{12} - g_3 \geq 0.$$

On the other hand, expert 2 has information about the financial cost of the sensors and he/she expresses this knowledge as $E2 = \{g_4, g_2, g_1, g_3\}$. The constraints from E2 are therefore,

$$g_4 - g_2 \geq 0,$$

$$g_2 - g_1 \geq 0,$$

$$g_1 - g_3 \geq 0.$$

Note, we expressed expert information as partial order inequalities. However, if exact numbers are know and if a user wishes to express them as specific FM values then this can be addressed as equality constraints. In this example it is apparent that all inputs are important and no single input is sufficient to solve this multi-criteria task. We desire a solution that has the fewest number of inputs and cost and is still sufficient for some problem/domain. It is also apparent that conflicts could easily arise between what is the fastest algorithm, the lowest cost sensor and what is the best for the training data.

Note, Example 1 naturally extends to challenges such as multi/hyper spectral signal processing in remote sensing or earth observation. Examples include multi-band and/or classifier feature- or decision-level fusion. The proposed GOAL programming and CI tools can also be used to address (via weighting) quality variation across data sets which arise due to factors like solar radiation, atmospheric conditions and time of year and weather in domains such as precision agriculture.

### 7.4 Experiments and Results

In this section, we demonstrate our work via synthetic experiments. Synthetic experiments are used to illustrate the behavior of the methods under different extreme and informative scenarios. We explore different extreme conditions instead of a single application. This is done in order to keep the problem tractable and the benefit is when we know the answer and all of the associated information then we can better study and understand how

the system reacts instead of explore what is otherwise for all intents and purposes a black box.

Training data was generated (in Matlab) $m$ pseud-randomly generated training instances with $N = 3$ inputs each. We use an *ordered weighted average* (OWA) with weights $\{0.7, 0.3, 0\}$ to generate our labels – a FM with values $g_1 = .7$, $g_2 = .7$, $g_3 = .7$, $g_{12} = 1$, $g_{13} = 1$, $g_{23} = 1$. The following scenarios involve different combinations of multiple expert inputs with varying weights and conflict. Table 7.1 reports the results for each of the following cases, which are outlined and discussed in detail in the following subsections.

Table 7.1
Learned FM for different scenarios involving training data, multiple experts and conflict.

| Scenario | weights | $g_1$ | $g_2$ | $g_3$ | $g_{12}$ | $g_{13}$ | $g_{23}$ |
|---|---|---|---|---|---|---|---|
| Training data only | training data = 1, expert = 0 | 0.7 | 0.7 | 0.7 | 1 | 1 | 1 |
| Data and expert with no conflict | training data = 0.3, expert = 0.7 | 0.7 | 0.7 | 0.7 | 1 | 1 | 1 |
| Priority on expert 1 (E1) and conflict | training data = 0.01, expert = 0.99 | 0.8709 | 0.8710 | 0.7335 | 0.8710 | 0.9601 | 0.8710 |
| Priority on training data and conflict | training data = 0.99, expert = 0.01 | 0.7696 | 0.9027 | 0.7204 | 0.9027 | 0.9820 | 0.9027 |
| Multiple experts (priority on E1) | training data = 0.1, E1 = 0.8, E2 = 0.1 | 0.7768 | 0.7768 | 0.7642 | 0.7768 | 0.9373 | 0.7784 |
| Multiple experts (priority on E2) | training data = 0.1, E1 = 0.1, E2 = 0.8 | 0.7660 | 0.7660 | 0.7644 | 0.7660 | 0.9377 | 0.7789 |

### 7.4.1 Training data only

In this scenario, row 1 from Table 7.1, we use a single training data set and we give it full priority, i.e., weight of 1. Thus, no expert information is used. We expect and do indeed recover the FM produced by our OWA. This scenarios is simple but it is informative because we see that we recover the desired answer and specifically we obtain the same result as if we just used the QP.

### 7.4.2 Single expert with no conflict

In this scenario, row 2 from Table 7.1, we use a single expert and training data. Specifically, no conflict between these sources exist. The expert provides the information $g_{23} \geq g_1$, which does not violate our monotonicity constraints. In this type of scenario, the expert is either reinforcing our monotonicity constraints (aka constraint redundancy) or they are giving us additional constraints (and insights) into which inputs are more important than others (which goes beyond just monotonicity). The key is, the training data and the expert are in agreement. We assigned a weighting of 0.7 and 0.3 for training data and expert respectively. As Table 7.1 shows, there is no change in the learned FM. Thus, as long as all of the criteria is in harmony, the resultant FM remains the same regardless of the distribution of weights on the expert and the training data.

### 7.4.3 Single expert and conflict

In this scenario, rows 3 and 4 from Table 7.1, we once again use a single expert and training data. However, we consider the case that conflict exists between the training data solution and our expert. Specifically, the expert gives us $g_1 \geq g_{23}$, which violates the criteria from the training data which values $g_{23}$ more than $g_1$. We can clearly see that, based on the relative weights, our resulting FM migrates between our training data only FM to the expert provided information (still in the context of the training data obviously). This is a difficult scenario to characterize in general. The expert knowledge is used, however the result is subject to our monotonicity constraints and ultimately the solution is driven relative to the SSE criteria.

### 7.4.4 Multiple experts

In this scenario, rows 5 and 6 from Table 7.1, we use multiple experts and training data. Specifically, let the criteria from the first expert be $g_1 \geq g_{23}$ and the second expert be $g_3 \geq g_{12}$. Thus, both experts have conflicting criteria between themselves as well as with the training data. In both scenarios, we let the importance of the training data set be relatively low and we vary the importance of the expert information. In the case that we place more weight on E1, we see that $g_1$ almost equals $g_{23}$. In the case that E2 is more important, we can see that $g_3$ almost equals $g_{12}$. They follow our input, however they are numerically similar because each solution (FM) is a complex combination of different weighted accounts of expert opinion and our training data. However, it is important to note that in each of the above cases our soft constraints were taken into account while our hard constraint, the boundary and monotonicity constraints, are enforced (otherwise we would not have learned a FM).

## 7.5 Conclusion and future work

Herein, we explored a new way to learn the CI for data/information fusion based on a combination of one more labeled training data sets and experts. Weighted Goal programming is a way to address natural conflicts that may arise within and across sources (e.g., data sets and experts). We showed that weighted Goal programming can be used to enforce soft (weighted) constraints and hard constraints, e.g., boundary and monotonicity constraints of the FM. Synthetic experiments were provided that showed a number of cases of different weighting schemes and forms of conflict.

In future work, we will explore how to include complexity goals for the FM in Goal programming, such as $L_1$ regularization. We will also explore proportion information if/when available, e.g., input $3$ is preferred by a factor of $2.4$ to input $1$. Last, we will apply this work to different real world problems, e.g., multi-sensor fusion and multi/hyper spectral processing.

# CHAPTER VIII

# DATA-DRIVEN COMPRESSION AND EFFICIENT LEARNING OF THE CHOQUET INTEGRAL

## 8.1 Introduction

Data/information fusion is an enabling theory for numerous fields, e.g., machine learning, signal/image processing, big data, *Internet Of Things* (IoT), bioinformatics, and cyber security, to name a few. In this paper, we focus specifically on aggregation as the term *fusion* has been elusive definition-wise (either too vague or overly specific). In general, the idea is to combine $N$ different inputs in such a way that the overall result (typically a reduction from $N$ inputs to one result) is somehow better than the outcome acquired using just the individuals by themselves. First, it is up to the user to define "better". For example, maybe the idea is to combine a set of inputs to create a single result that can be more easily visualized. The idea could also be to reduce (summarize) data so it is more manageable. In machine learning, better may mean achieving more generalizable decision boundaries for classifiers. The point is, "better" is a concept that needs to be specified relative to some task at hand. Next, focus shifts to how to combine these $N$ inputs. To date, most mathematical approaches have focused on combining inputs relative to the assumption of independence between them (which is advantageous tractability-wise). However, often there are rich interactions (e.g., correlations) between inputs that should be exploited. But for $N$ inputs,

there are $2^N$ possible subsets to consider. As $N$ grows, tractability is of utmost concern. The focus of this paper is a new tractable way to identify, model and exploit non-redundant data supported interactions. The ideas are presented at an abstract level as to not muddle the theory with any one particular application.

Herein, we focus on the *fuzzy integral* (FI) as it is a powerful and flexible aggregation function capable of exploiting rich interactions between inputs. In $1972$, Sugeno introduced the *fuzzy measure* (FM) (a normal capacity) in the context of the *Sugeno integral* (SI) [175]. Though Sugeno coined the term FI for his SI, the term FI has been generalized to a wider class of integrals. One well-known example is the *Choquet integral* (ChI), originally proposed by Gustav Choquet in $1953$ [44]. While the ChI was initially used in statistical mechanics and potential theory, in particular with respect to an additive probability measure, it has since found application in numerous other areas, e.g., computer vision [135, 181], classification [43, 77, 80, 72, 137, 203], pattern recognition [118, 27, 62], *multi-criteria decision making* (MCDM) [70, 127, 42, 185, 16, 17, 15], control theory, forensic science [8], Choquistic regression [183, 184], and *multi-kernel learning* (MKL) for *support vector machine* (SVM) classification and regression [152]. Numerous algorithms have been put forth to learn the FI from data, e.g., *quadratic programming* (QP) [76], gradient descent [121], penalty/reward [120], Gibbs sampler [136], and linear programming [26], to name a few.

For $N$ inputs, there are $2^N$ FM variables and $N(2^{N-1} - 1)$ monotonicity constraints. An advantage of the FI is we can model and exploit such knowledge. However, a drawback is lack of tractability. In practice, this is an important and often limiting factor. The

community has been exploring ways to solve this dilemma. The traditional approach is to require or learn just the densities (measure on just the singletons) and an imputation function is used to *fill in* the remaining variables, e.g., Sugeno characteristic polynomial and resultant $\lambda$-measure [182]. In a different approach, Grabisch defined the $k$-order additive FM/FI [71]. The $k$-additive FM/FI is a restriction limiting interactions to at most $k$ inputs. It can do this because the Mobius values for sets (variables) larger than $k$ are zero. Exploiting this property enables us to discard FM variables and obtain a lossless compression. The $k$-order additive measure is indeed efficient when $k$ is much less than the number of inputs. In many applications, e.g., MCDM, this often proves to be sufficient and of great utility. However, in other situations, e.g., pattern recognition, machine learning, signal/image processing, and computer vision, to name a few, we are often not dealing with humans per se versus automation and notions like bounded rationality do not apply. It can, and often is, the case that higher-order interactions exist and are crucial. Last, even if we can determine the order $k$, many problems render $k$-additivity ineffective. For example, the heavily utilized minimum and maximum aggregation operators–and other *linear combinations of order statistics* (LCOS) at that–are of $N$-order, requiring all $2^N$ MT terms for the FM and FI computation. Consequently, there is no savings in the number of variables. On the contrary, the computational complexity increases significantly, as the computation of the FI based on the MT is highly dense (needs all $2^N$ terms) verses the conventional formulation of the FI (which needs only $N$ terms). In closing, density-based imputation and $k$-order additivity have been explored to date and are applicable and of great use for different problems/contexts.

Herein, we propose a new approach that scales to the problem size by adapting to available training data. Our approach has four novel parts. First, data supported variables are identified and used in optimization. Identification of such variables also empowers us to know about the existence of future ill posed input scenarios; i.e., FI aggregations involving variable subsets that could not be inferred from data and therefore we should question. Second, we outline an imputation function framework to address data unsupported variables. Third, we present a lossless way to compress redundant variables and associated monotonicity constraints. This is important with respect to computation, memory storage and optimization. Last, we outline a lossy approximation method to further compress the ChI (if/when desired). In summary, our approach is different in philosophy from density-based imputation and $k$-additivity. However, our approach can be used to enhance $k$-additivity if the goal is to learn it from data.

This work is driven by the fact that a single *instance* of the FI for $N$ inputs uses only $N$ of the $2^N$ FM variables. Hence, we can learn at most $\min\{2^N, M \times N\}$ variables for a problem with $M$ observations and $N$ inputs. We know from linear algebra that in order to obtain unique solutions for $2^{20}$ FM variables in a $20$ input problem, we need at least $2^{20}$ independent observations, which is more than one million observations. While a problem with input sizes around $20$ is common, it is rare to find this huge number of samples for that problem. Now suppose the problem has $1,000$ observations, then at most $(20 \times 1,000) = 20,000$ variables are needed, when each observation is associated with unique set of variables, to represent the FI for all the observations. In reality, the actual number of variables is far fewer because many observations share variables among them. Another

significant advantage from using only the data-supported variables is that it reduces the monotonicity constraints substantially, which is exponential on the inputs for the full FI and, therefore, can be a limiting factor in many solvers. A sophisticated solver could possibly incorporate as many as $20,000$ variables, however, handling more than one million variables along with exponential order of constraints becomes near impossible for any kind of modern day solver and computing platform.

Last, before we delve into related work and new methods, an explanation of why we explore the QP for optimization is given. One of the most commonly encountered error functions in practice is the *sum of squared error* (SSE). This captures how different our learned target function is to a ground truth. However, other error and/or associated penalty functions can and have been used relative to learning aggregation operators. For example, in [37] Bustince et al. discussed problems related to definitions of penalty functions in the context of data aggregation. They gave examples of continuous penalty functions based on spread measures including standard deviation and variance and discussed the idea to define a penalty function for non-monotonic aggregation function. In [12], we investigated $\ell_p$ norm regularization to balance function error with minimum complexity FMs. The message is, herein we focus on a generic four step process for data-driven FI learning. The concepts put forth can be used by different solvers, e.g., particle swarm optimization, genetic algorithms, etc., based on a user's desired error and/or penalty function. Our focus is the four steps, not a particular solver.

The remainder of this paper is arranged as follows. Section 8.2 describes how to learn the FI, specifically the ChI, from data that includes the full set of FM variables. Section 8.3

is an example of a small problem to illustrate how the proposed method works. Section 8.4 details the proposed new methods followed by experiments and analysis in Section 8.5.

## 8.2 Background

In this section, the ChI is defined and its QP-based optimization is outlined. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of finite elements, which can be things like sensors, experts, criteria or attributes in decision making, or algorithms in pattern recognition. A discrete (finite $X$) FM is a monotonic set-valued function defined on the power set of $X$, $2^X$, as $\mu : 2^X \to \mathbb{R}^+$ that satisfies

(i) Boundary condition: $\mu(\emptyset) = 0$,

(ii) Monotonicity: if $A, B \subseteq X$ and $A \subseteq B$, $\mu(A) \leq \mu(B)$.

Often an additional constraint is imposed on the FM to limit the upper bound to 1, i.e., $\mu(X) = 1$. Throughout this paper, we consider this condition for simplicity and convenience, which is useful in contexts like decision-level fusion.

Consider a training data set containing $M$ pairs of observations and labels, i.e., $O = \{(\mathbf{o}_j, y_j)\}, j = 1, 2 \ldots, M$, where $\mathbf{o}_j \in \mathbb{R}^N$ is the $j$th observation, $y_j \in \mathbb{R}$ is the associated label, and $o_j(x_k)$ corresponds to the observed value for $j$th instance and $k$th input. Let $\mathbf{u} = [\mu(\{x_1\}), \mu(\{x_2\}), \ldots, \mu(X)]^T$ be the $2^N - 1$ dimensional vector of FM variables except $\mu(\emptyset)$. The discrete (finite $X$) ChI on $\mathbf{o}_j$ with respect to the FM $\mu$ is

$$C_\mu(\mathbf{o}_j) = \sum_{i=1}^{N} [o_j(x_{\pi_j(i)}) - o_j(x_{\pi_j(i-1)})] \mu(S_{\pi_j(i)}), \tag{8.1}$$

where $\pi_j$ is a permutation function for observation $\mathbf{o}_j$ on the indices that satisfies $0 \leq o_j(x_{\pi(1)}) \leq \ldots \leq o_j(x_{\pi(N)})$, where $S_{\pi_j(i)} = \left\{x_{\pi_j(i)}, x_{\pi_j(i+1)}, \ldots, x_{\pi_j(N)}\right\}$ and $o_j(x_{\pi_j(0)}) = 0$ [30]. Eq. (8.1) can be written in matrix form as $C_\mu(\mathbf{o}_j) = \mathbf{c}_j^T\mathbf{u}$, where $\mathbf{c}_j$ is a column vector containing the $(2^N - 1)$ coefficients for observation $\mathbf{o}_j$. Let $k$ be the index of variable $\mu(B \in 2^X)$ in $\mathbf{u}$. Then the $k$th element of $\mathbf{c}_j$ is $c_{jk} = o_j(x_{\pi_j(l)}) - o_j(x_{\pi_j(l-1)})$ if $\exists S_{\pi_j}(l) = B,\ l \in \{1, \ldots, N\}$, and 0 otherwise. The FM monotonicity constraints can be written as $\mu(A) \leq \mu(B), \forall A, B \subseteq X$ and $A \subseteq B$. The set of monotonicity constraints defined by the above inequality relations are exhaustive; however, there are many redundant constraints among them which can be excluded for an optimization problem. For example, if we include $\mu(\{x_1\}) \leq \mu(\{x_1, x_2\})$ and $\mu(\{x_1, x_2\}) \leq \mu(\{x_1, x_2, x_3\})$ as monotonicity constraints, then they also imply that $\mu(\{x_1\}) \leq \mu(\{x_1, x_2, x_3\})$, and there is no need to explicitly define all the relations. The minimal set of constraints for an FM is $\mu(A) \leq \mu(A \cup q), \forall A \subset X$ and $\forall q \in X, q \notin A$.

The SSE between the ChI for all the observations in the training data, $O$, and our labels is [76, 75]

$$\begin{aligned} E(O, \mathbf{u}) &= \sum_{j=1}^{M} (C_\mu(\mathbf{o}_j) - y_j)^2 = \sum_{j=1}^{M} (\mathbf{c}_j^T\mathbf{u} - y_j)^2 \\ &= \sum_{j=1}^{M} (\mathbf{u}^T\mathbf{c}_j\mathbf{c}_j^T\mathbf{u} - 2y_j\mathbf{c}_j^T\mathbf{u} + y_j^2). \end{aligned} \tag{8.2}$$

Based on this, the least square minimization problem can be expressed as [76, 75]

$$\text{(OP1)} \ \min_{\mathbf{u}} f_O(\mathbf{u}) = \mathbf{u}^T H\mathbf{u} + \mathbf{d}^T\mathbf{u},$$

$$\mu(A) \leq \mu(A \cup q), \quad \forall A \subset X \text{ and } \forall q \in X, q \notin A,$$

(monotonicity conditions) (8.3a)

$$\mu(\emptyset) = 0, \qquad \text{(boundary conditions)} \tag{8.3b}$$

$$\mu(X) = 1, \qquad \text{(normality conditions)} \tag{8.3c}$$

$$H = \sum_{j=1}^{M} \mathbf{c}_j \mathbf{c}_j^T \text{ and } \mathbf{d} = -2 \sum_{j=1}^{M} y_j \mathbf{c}_j.$$

## 8.3 Example 1: Data Supported Variables

In this section, we provide a simple numeric example of the underlying principle of the proposed method for a simple three input case ($N = 3$). Figure (8.1) shows the true underlying FM. Table 8.1 is an example training data set, $O$, with 5 instances and labels, drawn randomly from the true underlying FM.

Table 8.1
**Example 1**: Training data-set for a three input case ($N = 3$).

| Training data ($O$) | | | $S_{\pi_j(i)}$ | | | Used variables—$\mathbf{u}_P$ | Unused variables—$\mathbf{u}_Q$ |
|---|---|---|---|---|---|---|---|
| Index ($j$) | Observations ($\mathbf{o}_j$) | Labels ($y_j$) | $i=1$ | $i=2$ | $i=3$ | | |
| 1 | 0.6 0.5 0.1 | 0.41 | $\{x_1\}$ | $\{x_1, x_2\}$ | $X$ | $\mu(\{x_1\}), \mu(\{x_2\}), \mu(\{x_3\}),$ $\mu(\{x_1, x_2\}), \mu(\{x_1, x_3\}),$ and $\mu(X)$ | $\mu(\{x_2, x_3\})$ |
| 2 | 0.4 0.3 0.8 | 0.58 | $\{x_3\}$ | $\{x_1, x_3\}$ | $X$ | | |
| 3 | 0.9 0.2 0.7 | 0.66 | $\{x_1\}$ | $\{x_1, x_3\}$ | $X$ | | |
| 4 | 0.5 0.6 0.3 | 0.48 | $\{x_2\}$ | $\{x_1, x_2\}$ | $X$ | | |
| 5 | 0.6 0.2 0.7 | 0.57 | $\{x_3\}$ | $\{x_3, x_1\}$ | $X$ | | |

Example 1 has seven variables, denoted by $\mathbf{u}$, and five instances (training data). According to Eq. (8.1), the ChI for each instance requires FM variables for three sets, $S_{\pi_j(i)},\ i = \{1, 2, 3\}$ (column four in Table 8.1). In Example 1, only six of the seven variables, denoted as $\mathbf{u}_P$, are encountered–shown in column five of Table 8.1. Let us denote the unused variable, $\mu(\{x_2, x_3\})$, as $\mathbf{u}_Q$. We can split the structure in Figure (8.1) into two based on the

variables, $\mathbf{u}_P$ and $\mathbf{u}_Q$ (Figure (8.2)). The variables $\mathbf{u}_P$ can be learned by solving an optimization problem with only $\mathbf{u}_P$ as there are five unique ChI equations for five variables, and $\mu(X)$ is constant. On the other hand, there is no ChI equation involving $\mathbf{u}_Q$, and its value can be anywhere in the valid range, which is obtained using $\mathbf{u}_P$ and the monotonicity constraints on $\mathbf{u}_Q$. An imputation function, discussed in detail later, can be employed to assign a specific value within the interval range.

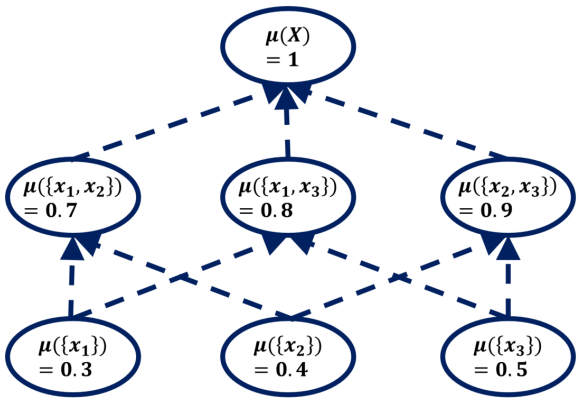



Figure 8.1

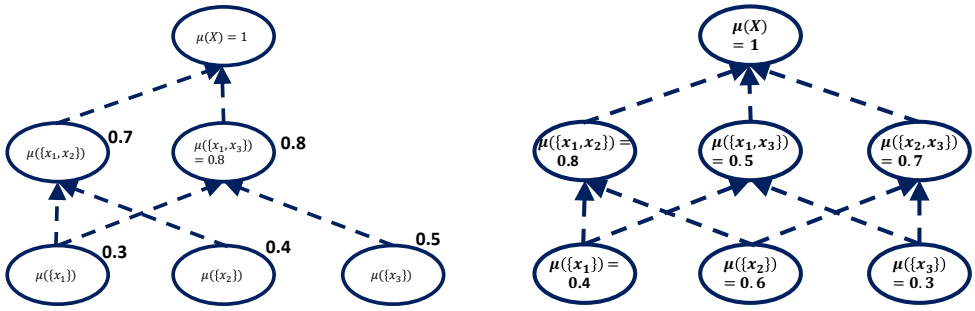
**Example 1**. Illustration of known (aka reference) FM for three inputs ($N = 3$). Nodes are variables and edges are monotonicity constraints.

## 8.4 Efficient ChI learning

### 8.4.1 Optimization with respect to just data supported variables

Based on our training data, we can partition the FM variables into two parts. The first set, $P \subseteq 2^X$, specifically $P = \left\{S_{\pi_j(i)}\right\}, \forall i \in \{1, 2, \ldots, N\}$ and $\forall j \in \{1, 2, \ldots, M\}$, is all variables that appear at least once in the ChI formula with respect to $O$ ($\mathbf{o}_i, i = 1, \ldots, M$) and the second set, $Q = 2^X \setminus P$, is all other variables. Let the cardinality of $P$ and $Q$ be $p$ and $q$ respectively. The partitioning of the power set leads to the decomposition of the

vector $\mathbf{u}$; $\mathbf{u} = [\mathbf{u}_P\ \mathbf{u}_Q]$, where $\mathbf{u}_P(l) = \mu(A),\ A \in P,\ l = \{1, 2 \ldots, p\}$, and $\mathbf{u}_Q(k) = \mu(B),\ B \in Q,\ k = \{1, 2 \ldots, q\}$. The coefficient vector $\mathbf{c}_j$ for each observation $\mathbf{o}_j$ is $\mathbf{c}_j = [\mathbf{c}_{Pj}\ \mathbf{c}_{Qj}]^T$, where $\mathbf{c}_{Pj}$ and $\mathbf{c}_{Qj}$ are the respective coefficient vectors of $\mathbf{u}_P$ and $\mathbf{u}_Q$ for the given observation. As the variables $\mathbf{u}_Q$ are not present in the ChI definitions of all the observations, their coefficients in the quadratic formula with respect to the training data are always zeros, i.e., $\mathbf{c}_{Qj} = \mathbf{0},\ \forall j \in \{1, 2, \ldots, M\}$.




(a) (b)

Figure 8.2

**Example 1**. (a) Illustration of required FM values for data in Table 8.1. Note, $\mu(\{x_2, x_3\})$ is not supported by training data and subsequently cannot be learned. (b) Illustration of data unsupported values and their interval-valued ranges due to monotonicity conditions. The values/intervals outside nodes signify that they are learned via optimization whereas those inside are used as constants.

The objective function in OP1 consists of a quadratic term with a coefficient matrix $H$ and a linear term with coefficient vector $\mathbf{d}$. These can be represented in terms of the

observation coefficients, and thus can be partitioned into *blocks*. The coefficient matrix, $H$, is therefore

$$\begin{aligned} H = \sum_{j=1}^{M} \mathbf{c}_j \mathbf{c}_j^T &= \sum_{j=1}^{M} \begin{bmatrix} \mathbf{c}_{Pj} \\ \mathbf{c}_{Qj} \end{bmatrix} [\mathbf{c}_{Pj}^T \quad \mathbf{c}_{Qj}^T] \\ &= \sum_{j=1}^{m} \begin{bmatrix} \mathbf{c}_{Pj}\mathbf{c}_{Pj}^T & \mathbf{c}_{Pj}\mathbf{c}_{Qj}^T \\ \mathbf{c}_{Qj}\mathbf{c}_{Pj}^T & \mathbf{c}_{Qj}\mathbf{c}_{Qj}^T \end{bmatrix} \\ &= \begin{bmatrix} 2\sum_{j=1}^{M} \mathbf{c}_{Pj}\mathbf{c}_{Pj}^T & \mathbf{0}_{PQ} \\ \mathbf{0}_{QP} & \mathbf{0}_{QQ} \end{bmatrix} \\ &= \begin{bmatrix} H_{PP} & \mathbf{0}_{PQ} \\ \mathbf{0}_{QP} & \mathbf{0}_{QQ} \end{bmatrix}, \end{aligned} \tag{8.4}$$

where $\mathbf{0}_{QP}$ is a $Q \times P$ matrix of all zeros and $H_{PP} = 2\sum_{j=1}^{M} \mathbf{c}_{Pj}\mathbf{c}_{Pj}^T$. Similarly, the coefficient vector $\mathbf{d}$ can be represented as

$$\mathbf{d} = -2\sum_{j=1}^{M} y_j \begin{bmatrix} \mathbf{c}_{Pj} \\ \mathbf{c}_{Qj} \end{bmatrix} = \begin{bmatrix} -2\sum_{j=1}^{M} y_j \mathbf{c}_{Pj} \\ \mathbf{0}_Q \end{bmatrix} = \begin{bmatrix} \mathbf{d}_P \\ \mathbf{0}_Q \end{bmatrix},$$

where $\mathbf{0}_Q$ is a $Q \times 1$ vector of all zeros and $\mathbf{d}_P = -2\sum_{j=1}^{M} y_j \mathbf{c}_{Pj}$. We can see from the alternate representation of the coefficient matrix $H$ in Eq. (8.4) that the diagonal blocks of the matrix are zeros. This indicates that variables $\mathbf{u}_P$ and $\mathbf{u}_Q$ in the quadratic terms are decoupled, and consequently the objective function can be represented as a linear sum of two functions with variables $\mathbf{u}_P$ and $\mathbf{u}_Q$. That is, $f_O(\mathbf{u}) = f_O(\mathbf{u}_P, \mathbf{u}_Q) = f_{O_1}(\mathbf{u}_P) + f_{O_2}(\mathbf{u}_Q)$.

Furthermore, the constraints can be divided into groups with respect to $\mathbf{u}_P$ and $\mathbf{u}_Q$. For sets $A, B \in 2^X$, we have the following four cases: (1) $\mu(A), \mu(B) \in \mathbf{u}_P$, (2) $\mu(A), \mu(B) \in$

$\mathbf{u}_Q$, (3) $\mu(A) \in \mathbf{u}_P, \mu(B) \in \mathbf{u}_Q$, and (4) $\mu(A) \in \mathbf{u}_Q, \mu(B) \in \mathbf{u}_P$. The boundary constraints can also be grouped based on: (1) $\mu(A) \in \mathbf{u}_P$ and (2) $\mu(A) \in \mathbf{u}_Q$. Now, we rewrite the optimization problem in OP1 in terms of $\mathbf{u}_P$ and $\mathbf{u}_Q$,

$$\begin{aligned}
\text{(OP2)} \min_{\mathbf{u}_P, \mathbf{u}_Q} f_O(\mathbf{u}_P, \mathbf{u}_Q) &= \mathbf{u}_P^T H_{PP} \mathbf{u}_P + \mathbf{u}_P^T \mathbf{0}_{PQ} \mathbf{u}_Q + \\
&\mathbf{u}_Q^T \mathbf{0}_{QP} \mathbf{u}_P + \mathbf{u}_Q^T \mathbf{0}_{QQ} \mathbf{u}_Q + \mathbf{d_P}^T \mathbf{u}_P + \mathbf{0}_Q \mathbf{u}_Q, \\
&= \underbrace{(\mathbf{u}_P^T H_{PP} \mathbf{u}_P + \mathbf{d_P}^T \mathbf{u}_P)}_{\geq 0, \text{ terms with only } u_P} + \\
&\underbrace{\mathbf{u}_P^T \mathbf{0}_{PQ} \mathbf{u}_Q + \mathbf{u}_Q^T \mathbf{0}_{QP} \mathbf{u}_P}_{=0, \text{ terms with both } u_P \text{ and } u_Q} + \underbrace{\mathbf{u}_Q^T \mathbf{0}_{QQ} \mathbf{u}_Q + \mathbf{0}_Q \mathbf{u}_Q}_{=0, \text{ terms with only } u_Q},
\end{aligned} \tag{8.5}$$

subject to: (1) $\mu(A) \leq \mu(B)$ for $\mu(A), \mu(B) \in \mathbf{u}_P$ and $A \subset B$, (2) $\mu(A) \leq \mu(B)$ for $\mu(A), \mu(B) \in \mathbf{u}_Q$ and $A \subset B$, (3) $\mu(A) \leq \mu(B)$ for $\mu(A) \in \mathbf{u}_P, \mu(B) \in \mathbf{u}_Q$ and $A \subset B$, (4) $\mu(A) \leq \mu(B)$ for $\mu(A) \in \mathbf{u}_Q, \mu(B) \in \mathbf{u}_P$ and $A \subset B$, (5) $\mu(A) \geq 0, \forall \mu(A) \in \mathbf{u}_P$, (6) $\mu(A) \geq 0, \forall \mu(A) \in \mathbf{u}_Q$, and (7) $\mu(X) = 1$, where $A, B \in 2^X$. It is obvious from OP2 that all the terms with $\mathbf{u}_Q$ in the objective function are zeros, and all the constraints involving both $\mathbf{u}_P$ and $\mathbf{u}_Q$ are inequality relations. That is, $\mathbf{u}_P$ does not depend on $\mathbf{u}_Q$, but rather the opposite. Therefore, we can optimize $\mathbf{u}_P$ first and then use its result to obtain values for $\mathbf{u}_Q$. As such, we break OP2 into two sequential tasks, OP2.1 and OP2.2, where

$$\text{(OP2.1)} \quad \min_{\mathbf{u}_P} f_{O_1}(\mathbf{u}_P) = \mathbf{u}_P^T H_{PP} \mathbf{u}_P + \mathbf{d}_P^T \mathbf{u}_P,$$

subject to $\mu(A) \leq \mu(B)$ for $\mu(A), \mu(B) \in \mathbf{u}_P$, $\mu(A) \geq 0$ for $\mu(A) \in \mathbf{u}_P$, $\mu(X) = 1$, where $A, B \in 2^X$.

The second step, OP2.2, is concerned with $\mathbf{u}_Q$ and is based on constraints defined by the $\mathbf{u}_P$ values learned in OP2.1,

$$\text{(OP2.2)} \min_{\mathbf{u}_Q} f_{O_2}(\mathbf{u}_Q) = \mathbf{u}_Q^T \mathbf{0}_{QQ} \mathbf{u}_Q + \mathbf{0}_Q \mathbf{u}_Q = 0,$$

subject to a valid FM in $[0, 1]$,

$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A), \mu(B) \in \mathbf{u}_Q, \tag{8.6a}$$

$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A) \in \mathbf{u}_P, \mu(B) \in \mathbf{u}_Q, \tag{8.6b}$$

$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A) \in \mathbf{u}_Q, \mu(B) \in \mathbf{u}_P, \tag{8.6c}$$

$$\mu(A) \geq 0 \text{ for } \mu(A) \in \mathbf{u}_Q, \tag{8.6d}$$

where $A, B \in 2^X$. It is worthwhile to note that OP2 includes the exhaustive set of monotonicity constraints and extended list of boundary constraints only to facilitate our partitioning of the inequality constraints so we can decompose OP1. Instead of enumerating all possible monotonicity conditions, we instead define the monotonicity constraints, e.g., Eq. (8.3a) in the standard QP formulation, with the minimal set of relations excluding all redundant constraints. In the same manner, the boundary conditions can be scaled down, reducing the number of constraints considerably in both OP2.1 and OP2.2.

Since OP2.2 is a $0$-valued objective function, it is in effect a constraint satisfaction problem that can be wrote as

$$\text{(OP2.2a) find } \mathbf{u}_Q \text{ subject to MC}(\mathbf{u}_Q) \tag{8.7}$$

where MC$(\mathbf{u}_Q)$ denotes the constraints in Eqs. (8.6a-d). The valid region defined by these constraints is a convex bounded polyhedron in a $q$-dimensional space denoted by

$C_Q$, where $q$ is the cardinality of $\mathbf{u}_Q$. Any point inside $C_Q$ is valid; therefore the whole convex polyhedron constitutes the solution set of the problem. Obviously, the problem has infinitely many solutions.

Our constraints can be further decomposed. Group I is unary constraints (Eqs. (8.6b-d)) since they include constants and variables from $\mathbf{u}_P$, which are themselves constants. In Group II, the constraints are binary with $\mathbf{u}_Q$ variables on both sides of the inequalities (Eq. (8.6a)). First, we consider the case of only Group I constraints. As these constraints specify that a variable is either less or greater than some constant, the hyperlines for these constraints run parallel to the axes of a $q$-dimensional space. The resultant solution set for $\mathbf{u}_Q$, $\tilde{C}$, is therefore a hyper-rectangle with $2^q$ vertices. Alternately, we can say that the valid range of each variable with regard to Group I constraints lies in an interval. Let $I(x) = [i_l(x), i_u(x)]$ be the interval for variable $x = \mu(A) \in \mathbf{u}_Q$, which can be deduced from Group I constraints as $i_l(x) = \max_{(B \subset A, \mu(B) \in \mathbf{u}_P)} \mu(B)$ and $i_u(x) = \min_{(B \supset A, \mu(B) \in \mathbf{u}_P)} \mu(B)$. It is trivial to show that if $\mu(E), \mu(F) \in \mathbf{u}_Q,$ and $E \subset F$, then $i_l(\mu(E)) \leq i_l(\mu(F))$ and $i_u(\mu(E)) \leq i_u(\mu(F))$ because each subset of $E$ is also a subset of $F$ and each superset of $F$ is also a superset of $E$. Substituting the unary constraints with intervals, OP2.2a can be rewritten as; find $\mathbf{u}_Q$ subject to (i) $\mu(A) \leq \mu(B)$ for $A \subset B$ and $\mu(A), \mu(B) \in \mathbf{u}_Q$ and (ii) $i_l(\mu(A)) \leq \mu(A) \leq i_u(\mu(A)), \mu(A) \in \mathbf{u}_Q$. The solution set, $C_Q$, is a subset of $\tilde{C}$ constrained by (i), which means that a valid point within $C_Q$ can be obtained from intervals by satisfying the monotonicity constraints on $\mathbf{u}_Q$.

In Section 8.4.3, we put forth an approach that allows data-unsupported variables to be computed on demand; hence we do not need to store or identify the valid region ex-

plicitly. That approach is based on the following concept. If $\forall A, B \in Q$, if $A \subset B$ then the interval calculated using Group I constraints always yields $I(A) \leq I(B)$ or $i_l(A) \leq i_l(B)$ and $i_u(A) \leq i_u(B)$. While $\tilde{C}$ can have a region over which $\mu(A) > \mu(B)$, the solution set, $C_Q$, does not contain any such region. Therefore, we can formulate a function, $f_I$, that maps an interval $I$ in $[0, 1]$ to a point in the interval, i.e., $f_I(I) \in I$, such that for any $I_i < I_j$, $f_I$ always preserves the relation $f_I(I_i) \leq f_I(I_j)$.

### 8.4.2 Computational Complexity

Here, we provide the computational complexity of the QP learning of data-supported variables. With training data of $M$ samples for an $N$ inputs problem, each observation can add at most $N$ variables, and the total number of training variables in the efficient ChI, $n_E$, can reach to $2^N-1$ variables at maximum for an overdetermined system. On the other hand, the number of training variables, $n_S$, in conventional ChI always is $2^{N-1}$ regardless of the training sample size. That is, $n_E \leq \min\{M \times N, 2^N - 1\}$ and $n_S = 2^{N-1}$. In a scenario when $M \times N < 2^{N-1}$, the worst case value for $n_E$ is, $n_E(\text{worst}) = M \times N < 2^{N-1} = n_S$. This implies that the worst case time complexity of the efficient ChI is less than the standard ChI when $M \times N < 2^N - 1$ (or $M < \frac{2^N-1}{N}$) and is equal when $M \geq \frac{2^N-1}{N}$, at which point the two methods converges to exactly the same problem with $2^{N-1}$ variables. In reality, the number of observations for large $N$ does not vary exponentially with $N$, but rather polynomially. For instance, we rarely encounter a $20$ input problem with $2^{20}$ observations, but we face problems whose number of observations can be modeled as $(c \cdot 20^x)$, where $c$ and $x$ are arbitrary constants. As the time complexity of a convex QP is on the order

of $O(k^3)$ for $k$ variables, our complexity is $O((MN)^3) = O(N^{3(x+1)})$, where $M = N^x$. That is, we have polynomial time complexity under the assumption that the number of observations is polynomial with respect to $N$.

### 8.4.3 Imputation of data unsupported variables

Section 8.4.1 showed that data supported variables in OP2.1 are scalars and that data unsupported variables lie in a convex bounded polyhedron defined by these scalars and the monotonicity conditions on the data unsupported variables. The focus of this section is a framework for assigning values (via an imputation function) to data unsupported variables based on the results of OP2.1 and *external knowledge*. First, we remark on a few important high-level considerations.

**Remark 5. Should I impute?** In all data-driven ChI learning work that we are aware of, optimization is with respect to data supported and unsupported variables. However, what is really being assigned to those data unsupported variables and should we "trust" future decisions (fusions) that rely on one or more of these data unsupported variables? Unlike prior work, this paper informs the reader about how to identify such ill-posed fusion scenarios. Beyond that, it is ultimately up to the user/system to decide what to do. One strategy is to not fuse. Another is to fuse but report that data unsupported variables were used. Herein, we explore the decision of fusing using data supported variables and a philosophy that lets us control unsupported variable value assignment, versus making it arbitrary or a side effect of the optimization algorithm.

**Remark 6. How is data-driven imputation different from density-driven imputation?** Technically, imputation is the mapping of interval-valued uncertainty to scalars for variables unsupported by data (or densities respectively). By definition, we are not privileged to know the "true" answer to data (or density) unsupported variables. This is the reality and type of problem with which we aim to advance. In density-based imputation, there are $2^N - 2 - N$ free (or *density unsupported*) variables. These variables have interval-valued uncertainty and a philosophy such as Sugeno's characteristic polynomial or a S-Decomposable measure is required for scalar assignment. In the context of our current paper, we are privileged to know (from data) much more than just the densities. Each observation *highlights* $N$ variables of increasing cardinality ($\mu(\{x_{\pi_j(1)}\}), \mu(\{x_{\pi_j(1)}, x_{\pi_j(2)}\}), ...$). Relative to density-driven, data-driven imputation reduces the number of and narrows the interval widths of data unsupported variables.

**Remark 7. How does data-driven imputation relate to $k$-additivity?** In $k$-additivity, a subset of variables–those whose cardinality is greater than $k$–are, or are forced to be, *irrelevant* relative to the problem at hand. What this means is no imputation occurs. However, in our data-driven approach variables are selected at each level of cardinality. No assumption is made with respect to if a problem is $k$-additive or not. Therefore, imputation is needed to fill in that which we do not know (that which is not observable). It is important to note that $k$-additivity and data-driven imputation are not "in competition". They are different tools. If a problem is $k$-additive and the desire is to learn it from data, then our four step approach can be combined with $k$-additivity.

Next, we outline an initial approach to data-driven imputation. However, imputation–deciding how to address that which we are not truly privileged to know–is a broad topic that is application/domain specific and the subject of numerous future works. Our intent here is to outline different initial approaches for the purpose of equipping the reader with options to explore relative to their task. We consider two different approaches. The first is where we specify the philosophy, e.g., expected value, optimistic or pessimistic. For example, an extreme case of a pessimistic philosophy would be to always select the lower interval value for each data unsupported variable interval. This imputation function models the belief that we are unwilling to assign any more utility to increasing subsets of inputs than what we observed in the data. The first approach requires the user to know something about the problem, to have some fundamental ideology that they wish to follow, or different approaches can be attempted and a winner selected. Our second initial idea is to take a machine learning approach like cross validation to help learn the imputation function.

**(Approach 1) Modeling:** Our geometric interpretation of the solution set for data unsupported variables informs us that if a function maps an interval to a scalar in that interval and if that function is also non-decreasing then it will always select a point in the valid solution space and the resultant FM will be monotone. Specifically, an imputation function, $f_I(I = [i_l, i_u])$, should possess the following properties; (i) $i_l \leq f_I(I) \leq i_u$ and (ii) the sub-gradients with respect to interval boundaries, $i_l$ and $i_u$, are non-negative for all intervals, $I$, i.e., $\frac{\partial f_I}{\partial i_l} \geq 0,\ \frac{\partial f_I}{\partial i_u} \geq 0$. The conundrum is, there is an infinite number of such real-valued functions. For sake of tractability, we explore three different types of functions

(optimistic, pessimistic and expected value like) that are convex combinations of our data unsupported variable interval endpoints,

$$f_{w_I}(I) = (1 - w_I)i_l + w_I i_u, \text{ s.t., } w_I \in [0, 1] \tag{8.8}$$

In the Appendix, we prove that Eq. (8.8) satisfies the conditions of an imputation function for a monotonic $w_I$ w.r.t $I$.

*(Case 1) Fixed:* In our first case, $w_I$ is a constant and $f_{w_I}(I)$ is therefore simply a linear combination of the interval endpoints. For example, if $w_I = 0$ we obtain $f_{w_I=0}(I) = i_l$. If $w_I = 0.5$, we take the expected value with respect to our observed (data supported) evidence. Furthermore, $w_I = 1$ is an optimistic assignment, i.e., $f_{w_I=1}(I) = i_u$.

*(Case 2) Dynamic:* In case 2, the idea is to not use a constant $w_I$. Instead, we select a "pivot point" (single value that characterizes our interval) according to $z_I(I) = (1 - \beta)i_l + \beta i_u$, where $\beta$ is a user/system constant. Next, $w_I$ is calculated using $z_I(I)$. The goal is to allow for a non-linear inflation and/or deflation based on the exact value of $z_I(I)$. For example, if $z_I(I)$ is "large" ("small"), e.g., $0.9$ $(0.1)$, then we might desire to inflate (deflate) our value versus what we linearly obtain in Case 1. One example is the Sigmoid (see Appendix for proof)

$$w_I(z_I(I)) = \frac{1}{1 + e^{-a(z_I(I)-b)}}, \tag{8.9}$$

where $a$ and $b$ are user/system parameters.

**(Approach 2) Machine learning:** Last, we discuss how $f_I$ can be learned from data by fitting the function to the learned variable, $\mathbf{u}_P$. There are at least two ways to accomplish this; (i) solve an optimization problem based on a criteria like the SSE relative to

the functions parameters or (ii) use a Hermite interpolation method to model this data with a monotonic piece-wise polynomial function [58]. The parameters of a dynamic weight function, $w_I$, can be learned from $\mathbf{u}_P$ following the below steps: First, determine the interval $I(x)$ for each variable, $x \in \mathbf{u}_P$ as if its actual value is unknown. That is, treat $x$ as a variable while others as constants with known values and use the monotonicity constraints along with values of $\mathbf{u}_P \setminus x$ to obtain the interval. Next, calculate $z_I$ according to $z_I(I(x)) = (1-\beta)i_l(x) + \beta i_u(x)$ relative to a fixed $\beta$. Then, compute $w_I$ (Eq. (8.8)) with the value of $z_I$ calculated in the previous step and $f_I(I)$ as the actual (learned) FM value for variable $x \in \mathbf{u}_P$. Last, fit a monotonic function of $w_I$ (e.g., sigmoid function in Eq. (8.9)) or piecewise monotonic polynomial function on the data $z_I$ vs. $w_I(I)$ to estimate its parameter. As the weight in the fixed imputation function can be modeled as $w_I(z_I) = c$, therefore, it can also be learned in the same manner described above. Note, bisection method can be used to select an appropriated value for $\beta$ in $z_I$ equation, however, a fixed value of $0.5$ was used in our experiments.

### 8.4.4 Lossless and lossy FM compression (variable elimination)

An imputation function does more than just help us build a real-valued FM. For same or approximate valued FM variables in $\mathbf{u}_P$, some variables can be removed and their exact or approximate values can be retrieved using the monotonicity constraints and imputation function.

Suppose the sets, for which the FM value is the same, say value $a$, comprises a family of sets, $V$, i.e., $\mu(v) = a, \forall v \in V \subset 2^X$. Now, assume that the sets, $\emptyset$ and $X$, are themselves

their subset and superset respectively. Then $V$ can be divided into three mutually exclusive families of sets:

1. $V_M$: Each element in $V_M$ has at least one superset and one subset in $V$,

$$\forall r \in V_M, \exists s, t \in V, s \subset r \subset t.$$

2. $V_L$: Each element in $V_L$ has at least one superset but no subset in $V$,

$$\forall r \in V_L, \nexists s \in V \subset r \text{ and } \exists t \in V \supset r.$$

3. $V_U$: Each element in $V_U$ has at least one subset but no superset in $V$,

$$\forall r \in V_L, \exists s \in V \subset r \text{ and } \nexists t \in V \supset r.$$

With respect to the above sets, the valid interval range for variables defined for each member set in $V_M$ can be derived from those in $V_L$ and $V_U$ respectively, and the interval will have the same lower and upper bounds, $a$, i.e., $\forall v \in V_M$, i.e., $\exists v_l \in V_L, \min(\mu(v)) = \mu(v_l) = a$ and $\exists v_u \in V_U, \max(\mu(v)) = \mu(v_u) = a$. The variables for $V_M$ can be removed, and their FM values can be recovered by any imputation function using the information only for $V_L$, $V_U$, and the monotonicity constraints. Next, without loss of generality, we illustrate FM variable compression for three cases.

**Min aggregation:** The FM values for the min aggregation operator for an $N$-input system are $\mu(A) = 0, \forall A$ where $A \subset X$, and $\mu(X) = 1$, where $V$ includes all the sets in the power set except $X$. $V_U$ includes only those sets with cardinality $N-1$, $V_U = \{B\}, \forall |B| = N-1$, and $V_L$ is empty. Therefore, $V_M = V \setminus (V_L \cup V_U)$, $|V_M| = 2^N - N - 1$,

and in total $2^N - N - 1$ variables can be removed. Figure 8.3(a) shows, as an example, the full set of FM variables for a 3-input system on the left side and the compressed FM variables on the right. Only four variables, for $V_L$ and $X$, are required for any imputation function to recover the remaining three. However, irrespective of problem size, the min imputation function needs only one variable, $\mu(X)$, which is by definition a constant.

**Max aggregation:** This operator for an $N$-input system is characterized as $\mu(A) = 0$, if $A = \{\emptyset\}$, else $\mu(A) = 1$, where $A \subseteq X$. With the same analysis as for the min, we get $V_L = \{B\}, \forall |B| = 1$, $V_U = \emptyset$, and $V_M = \{B\} \cup \emptyset, \forall |B| = 1$, which suggest that only $N + 1$ variables for the $N$ singletons and the empty set, $\mu(x) = 1$ where $x \in X$, are needed for computing the ChI of any observation. Figure 8.3(b) depicts the variable compression process for a 3-input max aggregation operator. We need to store only four variables, which can be further reduced to one when max imputation function is used.

**Binary FM:** An example of an arbitrary FM is shown in Figure 8.3(c). Of eight variables, four are 0-values and rest are 1-valued. Based on their values, these variables are partitioned into two clusters, $V_1$ and $V_2$, where $V_1$ contains 0-valued variables and $V_2$ has 1-valued variables. Using any imputation function, four variables can be removed, two from each cluster.

The above examples demonstrate the fact that we can reduce the number of variables significantly in a lossless way when a group of variables share the same value. In application, due to finite floating precision in computation, or simply to further reduce the

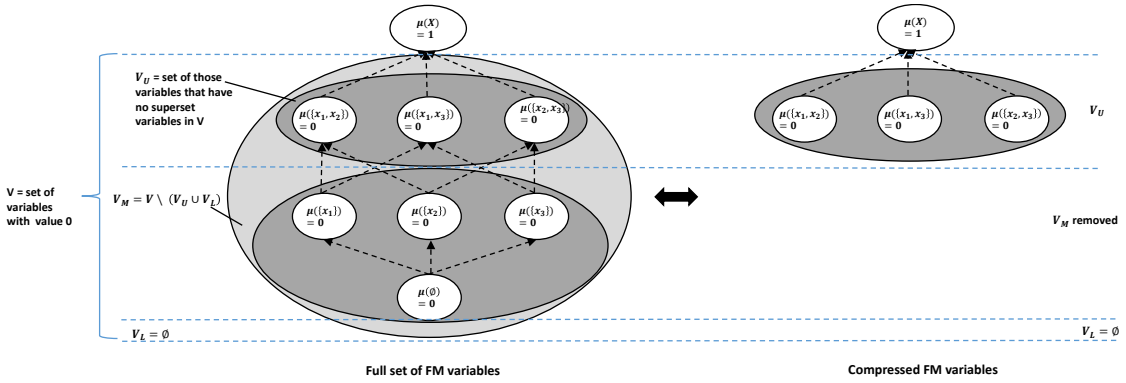
number of required variables, we might want to incorporate a tolerance ($\epsilon$) while removing variables, i.e.,

$$\max(\mu(v)) - \min(\mu(v)) \leq \epsilon, \forall v \in V, \text{ and } \epsilon > 0. \tag{8.10}$$

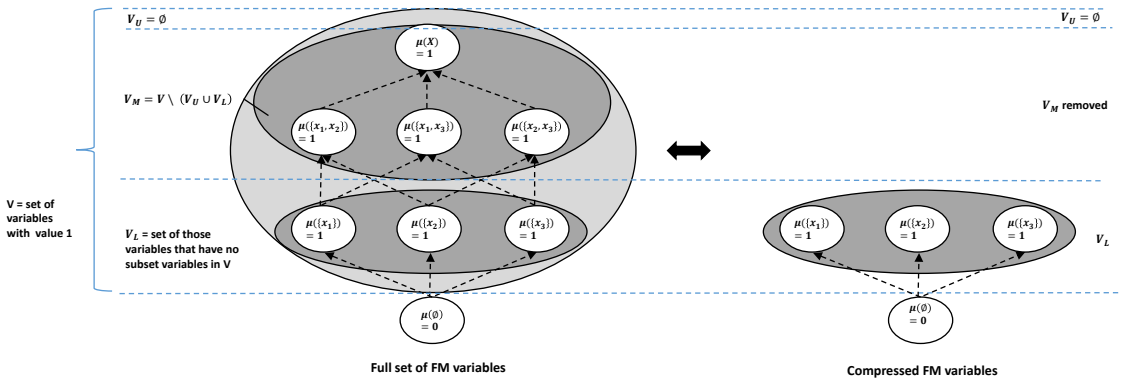
Note, Equation (8.10) is a lossy operation that yields ChI error.

## 8.5 Experiments

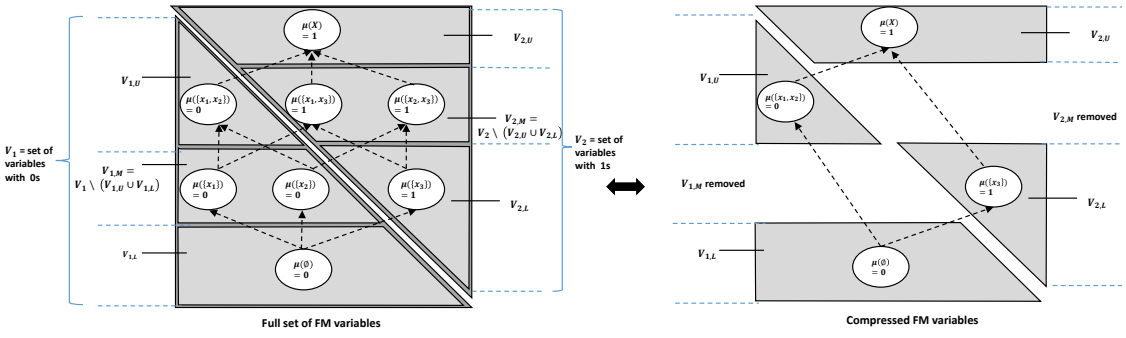
The aim of this section is to conduct controlled experiments, meaning we know the answer and can therefore precisely study different conditions, and to also compare the proposed method to relevant existing work. We do not use "real data sets," e.g., benchmark machine learning data sets, because we do not know what the "true" required aggregation strategy is or if there is even one (i.e., maybe a single input is sufficient). Instead, we focus on synthetic experiments because they allow us to explore a wider and richer range of conditions to demonstrate, confirm and learn about the theory put forth. Their results also therefore obviously trickle down into associated applications, e.g., computer vision, MKL, MCDM, etc. The following four experiments are performed. First, we investigate the impact of using only the data supported FM variables for learning versus using all FM variables. Second, we highlight similarities and differences to $k$-additivity. Third, we demonstrate how the proposed method can be used to efficiently represent and learn a relatively large scale problem (meaning otherwise considered intractable with respect to most modern computing platforms), $N = 20$. Last, we perform an experiment to show the impact of further compressing the FM by grouping similar valued FM variables.




(a)




(b)




(c)

Figure 8.3
Example of lossless compression (redundant variable elimination) of the FM for (a) min, (b) max, and (c) arbitrary binary FM for $N = 3$ system.

Data herein is generated (pseudo-)randomly from a uniform distribution. In experiments 1, 2 and 3, $N = 10$ is used, which yields $1,023$ FM variables and $5,110$ monotonicity constraints. The reason for $N = 10$ is because it is tractable (computationally and memory storage wise) by most solvers on modern day general purpose hardware. However, $N$ larger than $10$ quickly becomes difficult to solve—or already is more-or-less intractable. However, we could obviously increase $N$ if high performance computing hardware is available and the trends that we report below transfer without loss of generality. In addition, picking a "loadable" $N$ allows us to compare our method to all of the k-additive solutions (different $k$'s) and the full FI using all FM variables (no restrictions). Five thousand samples were generated. All but the fourth experiment is ran for training sample sizes of $15,\ 30, 75,\ 150,\ 300,\ 1000,$ and $3000$ to show important trends associated with different sample sizes. For each sample size, the training samples are selected (pseudo-)randomly from the data set of five thousand, the FM values are learned and then tested on the remaining observations. First three experiments are repeated $100$ times while the last experiment is repeated $10$ times for different selections of training samples, and the average is taken with respect to our performance metrics, the *mean of squared error* (MSE) (plotted on logarithmic scale in the figures) and the number of training variables used. Three thousand were selected specifically because it ensures that the system is overdetermined and has enough data to learn accurately all the variables in training (relative to $N = 10$). Three OWAs are used to generate the labels for the data; soft-max, mean and soft-min (OWA weights provided in Table 8.2). These OWAs were selected as they sample the "aggregation spectrum"—an optimistic operator, pessimistic operator and the last is an expected

value operator. Furthermore, these specific operators highlight interesting scenarios relative to $k$-additivity.

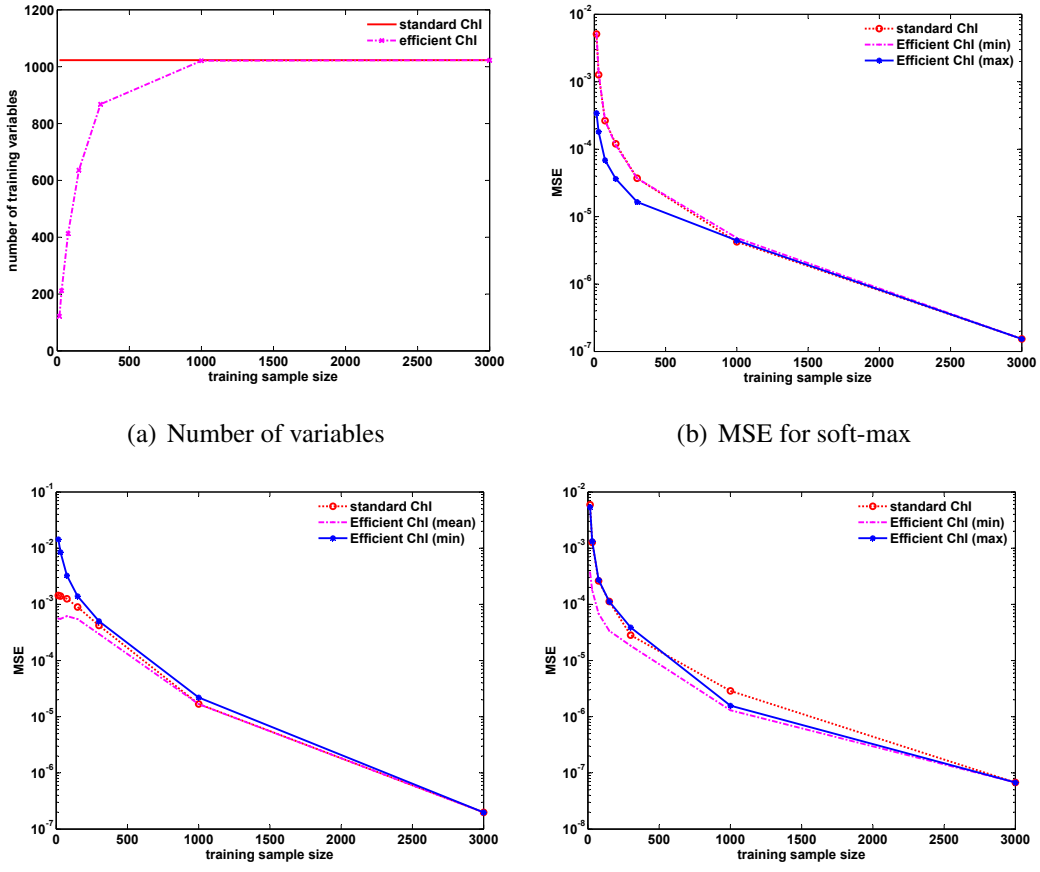
Table 8.2
OWA weights used in experiments.

| N | FM | OWA | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | Soft-max | 0.70 | 0.15 | 0.08 | 0.04 | 0.02 | 0.01 | 5E-3 | 2E-3 | 1E-3 | 6E-4 | | | | | | | | | | |
| | Mean | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | | | | | | | | | | |
| | Soft-min | 6E-4 | 1E-3 | 2E-3 | 5E-3 | 0.01 | 0.02 | 0.04 | 0.08 | 0.15 | 0.70 | | | | | | | | | | |
| 20 | Soft-max | 0.70 | 0.15 | 0.08 | 0.04 | 0.02 | 0.01 | 5E-3 | 2E-3 | 1E-3 | 6E-4 | 3E-4 | 1E-4 | 7E-5 | 4E-5 | 2E-5 | 9E-6 | 5E-6 | 2E-6 | 1E-6 | 6E-7 |
| | Mean | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| | Soft-min | 6E-7 | 1E-6 | 2E-6 | 5E-6 | 9E-6 | 2E-5 | 4E-5 | 7E-5 | 1E-4 | 3E-4 | 6E-4 | 1E-3 | 2E-3 | 5E-3 | 0.01 | 0.02 | 0.04 | 0.08 | 0.15 | 0.70 |

### 8.5.1 Experiment 1: Optimization with all variables vs. only data supported variables

In Experiment 1, we investigate the impact of using all FM variables versus just data supported FM variables. This allows us to explore two concepts. First, how much of a reduction are we really talking about in terms of numbers of variables? Second, experimentally, how much *variation* are we talking about with respect to imputation function selection versus learning with respect to all variables? Figure (8.4) shows the average number of training variables encountered and the MSE for soft-max, mean and soft-min. First, when there is sufficient data, e.g., $3,000$ training samples for $N = 10$, the standard and efficient ChIs (rightfully so) converge. Second, imputation enhances performance for undersampled problems when an appropriate function is selected. Third, the error amounts are low relative to our noteworthy savings in number of variables. It is important to highlight that while Experiment 1 shows that it is possible to achieve a good reduction in the

number of FM variables, and associated monotonicity constraints–which leads to much needed improvements in optimization time–the exact amount of savings depends on data volume and variety. This is what we should expect since our method is a way to focus on efficiency relative to data supported variables and accountability with respect to what we do not know (imputation function).




(a) Number of variables

(b) MSE for soft-max

(c) MSE for mean

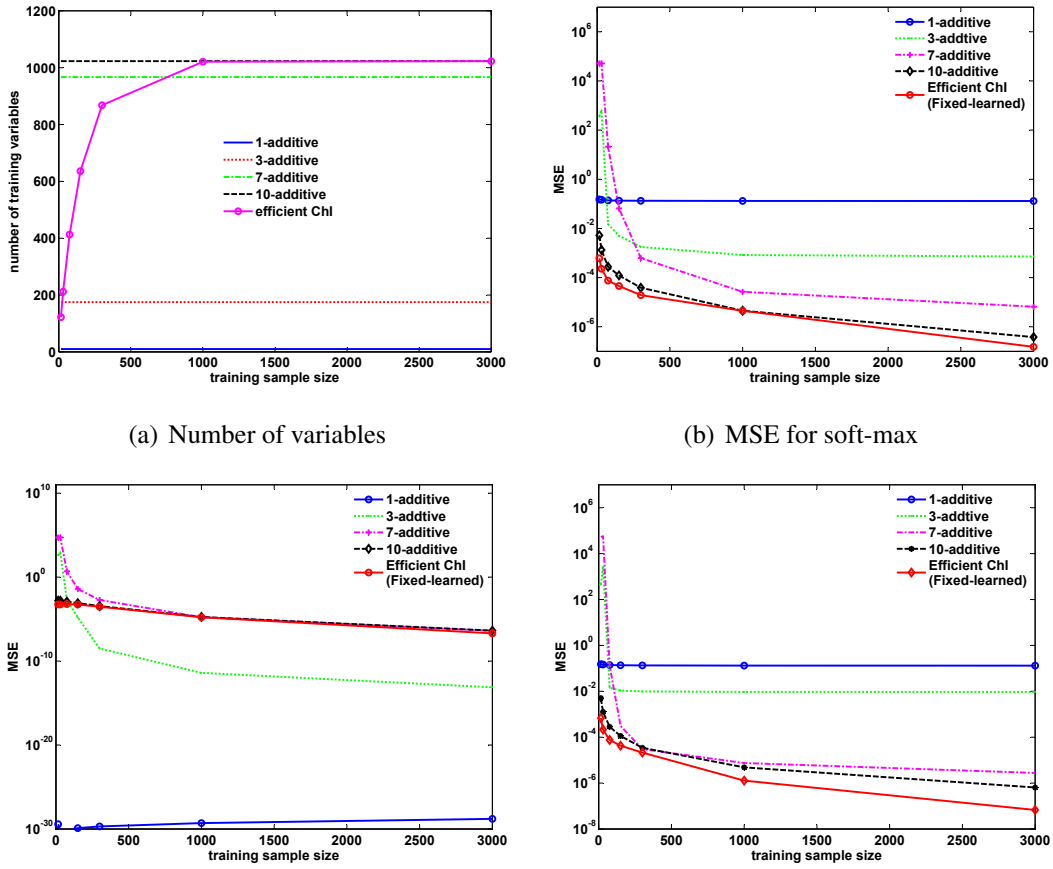
(d) MSE for soft-min

Figure 8.4
Experiment 1. Comparison between the standard and efficient computation of the ChI.

### 8.5.2 Experiment 2: $k$-additivity

Experiment 2 is not a comparison per se as our approach is data-driven, $k$-additivity is a matter of representation and ultimately the two can be combined into a single approach (if desired). In Experiment 2, we report different selections of $k$ relative to the underlying FMs. However, that is a lot of results and it becomes difficult (cluttered) to report in a table or figure. Therefore, without loss of generality, we restrict our analysis to $k = 1, 3, 7, 10$, which shows the trends.

Figure (8.5) tells the following story. For situations where $k$ is relatively (with respect to $N$) small and a good *fit*, e.g., Figure (8.5)(c) (the 1-additive mean operator), $k$-additivity does a wonderful job. However, when $k$ is large, e.g., Figure (8.5)(b) and Figure (8.5)(d) (soft max and soft min), our approach does particularly well with respect to both MSE and the number of required variables. Last, at three thousand samples both the 10-additive and our data-driven ChI correspond (turn into) the full ChI. Note, the 10-additive ChI has slightly higher MSE—which is very low for both, $1.5 \times 10^{-6}$ for the efficient Chi and $3.8 \times 10^{-7}$ for the $k$-additive. The reason is due to dense computation of the ChI in $k$-additivity– which can have as many as $2^N - 1$ non-zero terms for $N$ inputs when full-additive is used versus at most $N$ non-zero terms for conventional ChI–the symmetric matrix $D$ in QP optimization problem can be much denser relative to the standard/efficient ChI. Moreover, constraints in $k$-additivity involves more non-zero terms than the standard/efficient ChI and includes a global boundary constraint that contains all variables, the sum of which must be one. On the other hand, the standard/efficient ChI contains only unary and binary constraints.




(a) Number of variables

(b) MSE for soft-max

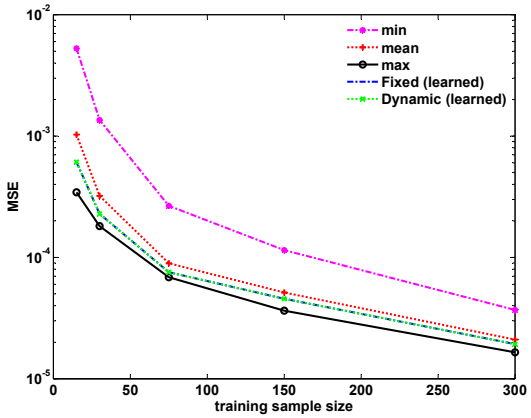
(c) MSE for mean

(d) MSE for soft-min

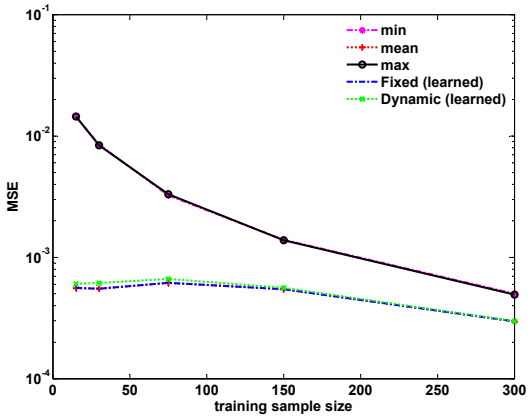
Figure 8.5
Experiment 2: Joint exploration of $k$-additivity and our data-driven ChI method.

### 8.5.3 Experiment 3: Imputation function exploration

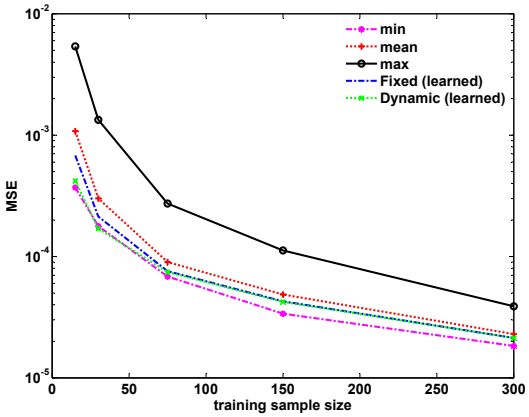
In Experiment 3 we investigate the impact of using different imputation functions. Specifically, we explore: (i) **min**: weight, $w_I = 0.0$; (ii) **mean**: $w_I = 0.5$; (iii) **max**: $w_I = 1.0$; (iv) **fixed learned**: $w_I = c$, where $c$ is learned via fitting the $w_I$ function; and (v) **dynamic learned**: $w_I(z_I(I))$ is a sigmoid function, i.e., $w_I(z_I(I)) = \frac{1}{1+e^{-a(z_I(I)-b)}}$ with parameters $a, b$ that are learned. We used a fixed value, $0.5$ for $\beta$, though it could be learned via the bisection method.




(a) MSE for soft-max




(b) MSE for mean




(c) MSE for soft-min

Figure 8.6
Experiment 3: Results relative to different imputation functions.

It is intuitive that the quality of the results (see Figure (8.6)) depends on how closely the imputation function matches the true underlying FM. We see that the max and mean have lower MSE than min for soft-max. Similarly, the mean imputation function is the best for the mean FM, and the min imputation function has better results for the soft-min FM. The fixed-learned imputation function performs the best for all cases as it can capture the trend exemplified by the variables. For example, the learned imputation function for the soft-max FM was a soft-max, which could not be obtained using our included set

of fixed imputation functions. The dynamic-learned imputation functions perform better than fixed but somewhat lags behind fixed-learned in two cases as the sigmoid function cannot exactly model a fixed weight function–which is the case for all three FMs. Using a more flexible function that can represent static functions could have better modeled the OWAs; however, there might be some cases for which sigmoid performs better than a fixed imputation function. Overall, a trade-off has to be made between simplicity and complexity while selecting an imputation function. Note, while using different imputation functions can definitely impact the efficient ChI's performance for relatively small sample sizes, their effects diminish as sample size increases because the number of variables needed to be imputed gradually decreases.

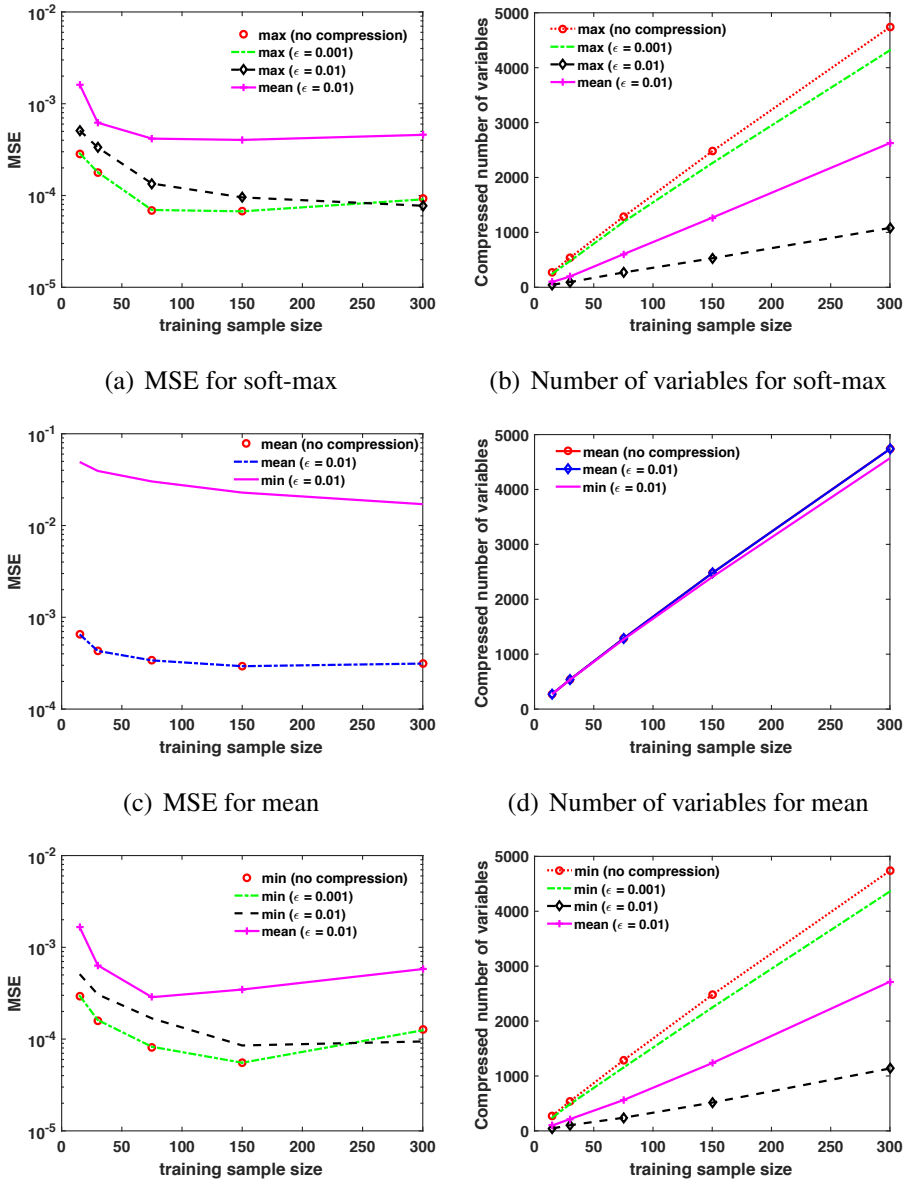
### 8.5.4 Experiment 4: (Relatively) Large $N$

In Experiment 4 we explore another payoff of the proposed work, the ability to extend the ChI to "bigger $N$". We selected $N = 20$, which already has $1,048,575$ variables and $10,485,740$ constraints. As stated earlier, one can obviously address $N > 20$ on better–high performance computing–hardware. Without loss of generality, the trends we expose and discuss next extend naturally to such an environment. Note, our efficient ChI with min imputation is the same as the standard ChI, which enables us to learn larger scale problems similar to the standard ChI but with a manageable number of variables. Performance can be greatly improved by using a suitable imputation function that aligns to the underlying FM.

The data set for Experiment 4 is (pseudo-)randomly generated from a uniform distribution. As described in Section 8.4.4, we perform variable compression to investigate the impact on the SSE and the number of subsequent variables. We consider two values for tolerances, $\epsilon = 0.01$ and $0.001$, which are used as thresholds for removing variables. For simplicity, instead of clustering variables with similar values, we checked the criteria defined in Eq. (8.10) for each individual variable and we removed it from the stored variables list if the criteria was satisfied. Figure (8.7) shows the results, where (a), (c), and (f) illustrate the impact that the imputation functions and tolerances have on the MSE whereas (b), (d), and (f) depict the impact on the number of variables. As expected, max imputation has lower error than mean for soft-max (Figure 8.7(a)), mean imputation function yields better results for mean (Figure 8.7(c)), and min imputation gives superior performance than mean for soft-min (Figure 8.7(e)).

As Figure (8.7) shows, imputation function selection impacts the MSE. For example, using max imputation has approximately four times improvement over mean for $1,000$ samples. On the contrary, changing the tolerances—with respect to the range under investigation—shows little-to-no impact, e.g., $\epsilon = 0.001$ provides the same results as no compression. While $\epsilon = 0.01$ has slightly worse MSE at lower training samples, it shows improved performance for relatively higher sample sizes (e.g., $1,000$). This may be because large numbers of samples generate huge numbers of variables, therefore removing relatively larger numbers of variables with closely related values and then imputing them with the same values can improve performance for the given soft-max and soft-min. Note, the soft-max has values close to one for variables residing at the upper layers in the lattice whereas

soft-min has values close to zero at the bottom layers. This specific phenomena is unique to these types of FMs.




(a) MSE for soft-max

(b) Number of variables for soft-max

(c) MSE for mean

(d) Number of variables for mean

(e) MSE for soft-min

(f) Number of variables for soft-min

Figure 8.7
Experiment 4: Efficient ChI results for $N = 20$ with and without variable compression for different tolerances.

Next, the efficient ChI uses on average $4,778$ training variables for $300$ samples, which is just $0.46\%$ of the total $2^{20} - 1$ variables. When compression is used, the min and max imputation functions provide maximum savings. For example, the max function with $\epsilon = 0.01$ reduces $4,778$ variables of the soft-max down to $1,081$ variables, a savings of $77.38\%$. In comparison to the max, the mean function uses a relatively higher number of variables $(4,569)$. However, for mean there is almost no gain in terms of variable compression (mean imputation even with $\epsilon = 0.01$ uses all variables whereas min uses just $1,136$ variables), which tells us the degree of compression depends obviously on both the FM and the type of imputation function used. As discussed in Section 8.4.4, the imputation function can offer savings only when the variables across multiple layers in the lattice have similar values, which is the case for soft-max and soft-min but not for mean, thus, providing no compression benefit for mean.

Last, Experiment 4 tells us that the MSE can reach an impressive rate (on the order of $10^{-3}$) at a relatively small sample/variable size. This "cut-off point" depends on the acceptable error rate as well as the complexity of the underlying FM. The results suggest that when we have limited resources, we may not want to run the efficient ChI on all the samples, but rather randomly sample a fraction of the data, thus keeping the number of variables at a workable level and still achieve acceptable performance for a less complex solution.

## 8.6 Conclusion

Herein, we introduced a new efficient and flexible data-driven way to represent and learn the FI. Specifically, variables supported by data are identified and used in optimization. Identification of data supported variables also allows us predict future ill posed input scenarios; ChI aggregations involving variable subsets that could not be inferred from data. Second, we outlined an imputation function framework for attacking data unsupported variables. Third, we presented a lossless method to compress redundant variables and monotonicity constraints. Last, we outlined a lossy approximation method to further compress the ChI (if/when desired). Optimization was cast in the context of quadratic programming. However, the four step process is independent of the underlying solver. Complexity analysis was provided and experiments were provided to allow more functional insight into the proposed theories. Advantages were shown relative to experiments in using all versus just data supported variables, $k$-additivity, impact of imputation function selection and computing the ChI for (relatively) large $N$. We noted that, like the $k$-additive integral and density-driven imputation, different applications have different needs and different amounts of knowledge are available. Overall, there does not seem to be a "best approach" but instead a set of tools for different problems/contexts.

In future work, we will focus on improving the imputation function. For example, we will try to develop intuition to help a user map attributes of their problem to a particular function (or family of functions). Furthermore, we focused on solutions that produced a real-valued number from interval-valued information. However, we have extensions for the FI, both integrand and FM, that allow for computation with respect to interval and

set-valued information. It will be interesting to instead of forcing a real-valued number to instead compute with the full uncertainty and discover what benefits, if any, there are in such an approach. We also plan to study how the use of an imputation function helps address overfitting. Furthermore, we partitioned the problem into data supported and data unsupported variables. However, within the set of data supported, not all variables are supported the same (number of samples). We would like to further study how to combat different degrees of "data supported" variables in learning. Next, we plan to explore different error and/or penalty functions and associated optimization algorithms. Since we learn from data and since the ChI has potentially many variables, over fitting can occur. The reader can refer to [12] for our prior work on regularization based learning of the ChI, which fits in naturally to the proposed paper and solver. However, it is likely that selection of imputation function can help as well. In future work, we will investigate questions like this to make the proposed work more generalizable. Last, the $k$-additive FI is a well-known and utilized approach in areas like MCDM, due to reasons like bounded rationality. In situations where it is desired and appropriate to learn a $k$-additivity solution, we will explore the impact of integrating our proposed four step data-driven learning to realize an improved technique; both in terms of data supported versus data unsupported variables but also the lossless and lossy compression of variables.

## Appendix

**Proposition 1:** Function $f_{w_I}(I) = (1 - w_I)i_l + w_I i_u$ is a valid imputation function if $w_I \in [0, 1]$ is non-decreasing with respect to interval $I = [i_l, i_u] \subseteq [0, 1]$.

Proof: It is sufficient to show that (i) $f_{w_I}(I) \in [i_l, i_u]$, and (ii) $f_{w_I}$ is non-decreasing with respect to interval $I \subseteq [0, 1]$.

i) As $f_{w_I}(I)$ is simply the linear convex sum of $i_l$ and $i_u$, $f_{w_I}(I) \in [i_l, i_u]$.

ii) The partial derivative of $f_{w_I}(I)$ with respect to the lower interval endpoint $i_l$ gives

$$\frac{\partial f_{w_I}(I)}{\partial i_l} = (1 - w_I) - \frac{\partial w_I}{\partial i_l} i_l + \frac{\partial w_I}{\partial i_l} i_u = (1 - w_I) + (i_u - i_l) \frac{\partial w_I}{\partial i_l}.$$

As this is a non-decreasing function, $w_I$ has a non-negative gradient, i.e., $\frac{\partial w_I}{\partial i_l} \geq 0$ and $w_I \in [0, 1]$. Therefore, $\frac{\partial f_{w_I}}{\partial i_l} \geq 0$ or non-decreasing with respect to $i_l$. Similarly, it can easily be shown that $\frac{\partial w_I}{\partial i_u} = w_I + (i_u - i_l) \frac{\partial w_I}{\partial i_u} \geq 0$, which concludes that $f_{w_I}$ is an imputation function. ■

**Proposition 2:** The function $w_I(I) = w_I(z_I(I))$, where $z_I = (1 - \beta) i_l + \beta i_u$ is a valid weight function in Eq. 8.8 if it has the following properties: (i) $\beta \in [0, 1]$, (ii) $h : [0, 1] \rightarrow [0, 1]$ and (iii) $h(z_I)$ is non-decreasing.

Proof: First, we show that $w_I$ lies in $[0, 1]$. Then, we prove that $w_I$ is a non-decreasing function w.r.t $I$.

(i) When $\beta \in [0, 1]$, $z_I(I)$ becomes the linear convex sum of interval endpoints, therefore $z_I(I) \in [0, 1]$. If $h$ holds the second property in Prop. 1, then $w_I(I)$ is in $[0, 1]$.

(ii) Taking partial derivative of $w_I$ with respect to $i_l$ gives

$$\frac{\partial w_I}{\partial i_l} = \frac{\partial w_I}{\partial z_I(I)} \frac{\partial z_I(I)}{\partial i_l} = (1 - \beta) \frac{\partial z_I(I)}{\partial i_l}.$$

Using Properties (i) and (iii), $\frac{\partial z_I(I)}{\partial i_l} \geq 0$. Following the same procedure, it can also be shown that $\frac{\partial z_I(I)}{\partial i_u}$ is non-negative. Therefore, $w_I$ is a valid weight function. Note that the weight function, $w_I$, based on sigmoid function has all the properties above. ■

# CHAPTER IX

# EFFICIENT BINARY FUZZY MEASURE REPRESENTATION AND CHOQUET INTEGRAL LEARNING

## 9.1 Introduction

Data/information fusion can be described as the intelligent combining of multiple sources/inputs to provide a more accurate, summarized, and/or reliable result than what a single source can achieve on its own. Driven by the need for better results, countless applications in many fields, such as computer vision and remote sensing, have long been applying fusion at different "levels" (signal, feature, decision etc.). Furthermore, the daily advancement in engineering technologies like smart cars, which operate in complex and dynamic environments using multiple sensors, are raising both the demand for and complexity of fusion.

While there is a multitude of *fuzzy integral* (FI) variants for fusion, we focus in this paper on the *Choquet Integral* (ChI), a well-known, demonstrated, and flexible aggregation function. The ChI has been used in numerous applications (mostly focused on decision level fusion), e.g., humanitarian demining [155], computer vision [181], pattern recognition [77, 80, 137, 118, 62], multi-criteria decision making [70, 127], control theory [187], and multiple kernel learning [152, 150, 155, 98, 97]. The ChI is a nonlinear aggregation function paramterized by the *fuzzy measure* (FM), a normal and monotone capacity. The

FM is defined on the power set of the sources, i.e., on the sets of all possible combination of sources, and therefore has $2^N$ variables for $N$ sources. With the flexibility of choosing values for these $2^N - 2$ parameters (excluding null set and $X$, which have fixed values by definition) in the FM, the ChI covers a wide range of aggregation operators. However, this advantage comes at a price: the requirement to specify (by human) or learn (from data) the FM. This means that the complexity of a learning problem, both in terms of storage and computation, is on the order of exponential to the number of sources. Therefore, a learning problem with the full set of variables becomes intractable at a relatively small $N$. Different approaches exist to learn the ChI from data, e.g., *quadratic programming* (QP) [76], gradient descent [121], penalty/reward [120], Gibbs sampler [136], linear programming [26], and efficient optimization with only data-supported variables [111].

In [9], Anderson et al. explored the *binary fuzzy measure* (BFM), a variant of the FM that takes values in $\{0, 1\}$ instead of $[0, 1]$. The BFM was motivated by the work of Du et al. in [55], in which the ChI was used to fuse information for a binary decision making problem relative to uncertainty from multiples sources. In particular, [55] extended the ChI for multiple instance learning in classification and showed that the learned FM had values approximately in $\{0, 1\}$ versus $[0, 1]$. This suggests that the underlying FM in some applications can be binary, which motivates the use of BFM directly rather than the generic FM due to its simplicity and efficient computation. Thus, many problems are a natural fit for the BFM and others are likely approximatable.

The BFM also has some nice properties and computational advantages over the FM. Anderson et al. showed that the ChI relative to BFM is equivalent to the Sugeno integral.

They also showed that only one variable is effectively used for the ChI computation of an observation in compared to $N$ variables for the real valued FM. Moreover, only one-valued variables need to be stored since zero-valued variables can be discarded. These features make the ChI computation and its storage (the BFM) less expensive compared to the real valued FM and ChI.

Herein, we first put forth an efficient data-driven learning method for the BFM and subsequently the BChI, which we refer to as *efficient BChI* (EBChI). Based on the fact that only one variable contributes to the BChI computation of an instance, we can explain this for variable selection during learning. Thus, each training instance adds at most one variable and the learning problem consequently becomes scalable to the problem size. That is, the number of variables to be optimized is no longer exponential of $N$, but rather linear to the number of instances (in the worst case). This not only lessens the computation burden, but it also provides a more robust and generalized solution since the number of unknowns (variables) are always fewer then the number of equations (training instances). In contrast, the learning problem with the entire set of FM variables is prone to overfitting as the number of training samples now becomes much smaller than the number of variables $(2^N - 2)$ [12].

Next, we provide a representation scheme for the BFM with the minimum set of variables, which we call *efficient BFM* (EBFM). The BFM variables can be partitioned into two groups, one-valued variables and zero-valued variables. Among the one-valued variables, some of them can deduce their values from others using the FM's monotonicity property (which we call dependent variables) while others cannot (which we call independent vari-

ables). The dependent variables can be eliminated without any loss of information and the BFM can be represented with only the independent variables, of which there can be at most $\binom{N}{N/2}$ variables. The full set of FM variables can be retrieved from these independent variables and vice-versa. Therefore, the independent variables constitute the minimal BFM or EBFM.

The remainder of this article is organized as such. In Section 9.2 we give the preliminaries of the FM, BFM, ChI, and BChI. Section 9.3 describes the efficient data-driven BFM learning followed by the representation in Section 9.4. In Section 9.5, we conduct controlled experiments on synthetic data to demonstrate the performance of our proposed learning method.

## 9.2 Fuzzy Measures and Choquet Integral

Let $X = \{x_1, x_2, ..., x_N\}$ be a set of $N$ inputs. A real-valued FM is a monotonic function defined on the power set of $X$, $2^X$, as $\mu : 2^X \rightarrow \Re^+$ that satisfies the following properties:

(i) (boundary condition:) $\mu(\emptyset) = 0$,

(ii) (monotonicity:) if $A, B \subseteq X, A \subseteq B$, $\mu(A) \leq \mu(B)$.

Often an additional constraint is imposed on the FM to limit the upper bound to 1, i.e., $\mu(X) = 1$.

Let $h(x_i)$ be the data/information from the $i$th input. The discrete ChI (finite $X$) is

$$\int_C \mathbf{h} \circ \mu = C_\mu(\mathbf{h}) = \sum_{i=1}^{N} h(x_{\pi(i)}) \left[\mu(S_{\pi(i)}) - \mu(S_{\pi(i-1)})\right], \tag{9.1}$$

where $\pi$ is a permutation of $X$, such that $h(x_{\pi(1)}) \geq h(x_{\pi(2)}) \geq \ldots \geq h(x_{\pi(N)})$, $S_{\pi(i)} = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$, and $\mu(S_0) = 0$. Equation (9.1) is often referred to as the difference-in-measure form since the integral is represented as the sum of difference-in-measure weighted by the input-values. The ChI can equivalently be written in difference-in-inputs form as

$$\int_C \mathbf{h} \circ \mu = C_\mu(\mathbf{h}) = \sum_{i=1}^{N} [h(x_{\pi(i)} - h(x_{\pi(i+1)})] \mu(S_{\pi(i)}), \tag{9.2}$$

where $h(x_{\pi(N+1)}) = 0$. The latter weighted-measure form at (9.2) is suitable for the FM learning problem herein, where $\mu$ is unknown. Equation 9.2 can be written in matrix form to facilitate optimization as

$$C_\mu(\mathbf{h}) = \mathbf{c}^T \mathbf{u}_B, \tag{9.3}$$

where $\mathbf{u}_B$ is the vector of all variables except $\mu(\emptyset)$ and has a length of $2^N - 1$, and $\mathbf{c}$ holds the coefficients of $\mathbf{u}_B$ for observation $\mathbf{h}$.

The FM can be visualized with respect to its uncertainty in a lattice (shown in Figure 9.1). Each instance yields a sort, $\pi$, which produces a walk up the lattice. The walk starts with $\mu(\emptyset)$ followed by $N$ other variables, each of different size cardinality. For example, an observation $\mathbf{h}$ with $h(\{x_2\}) \geq h(\{x_1\}) \geq h(\{x_4\}) \geq h(\{x_3\})$ walks along the path shown in Figure 9.1(b) and the corresponding ChI has variables $\mu(\{x_2\}), \mu(\{x_1, x_2\})$, $\mu(\{x_1, x_2, x_4\})$, and $\mu(X)$.

### 9.2.1 The Binary Fuzzy Measure

As already stated, a BFM, $\mu_B$, is a special case of the real-valued FM, $\mu$, that restricts $\mu_B \in \{0, 1\}$ instead of $[0, 1]$. Obviously, this drastically reduces the search space for an

optimization problem. In article [9], we proved that the BChI and Sugeno Integral are equivalent. The BChI is simply the standard Choquet Integral with respect to the BFM. Suppose the BFM values along the walk for $\mathbf{h}$ are given by

$$\mu_B(S_{\pi(i)}) = \begin{cases} 0 & \text{if } i < k \\ \\ 1 & \text{else,} \end{cases} \tag{9.4}$$

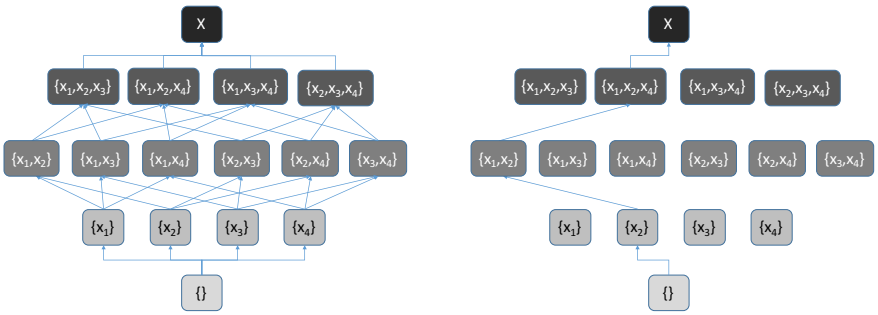
where $\mu_B(S_{\pi(k)})$ is the first variable encountered along this path with value 1. Replacing the FM with this BFM in Eq. (9.1) and then expanding it, the BChI can be written as [9]

$$\begin{aligned} \int_C \mathbf{h} \circ \mu_B &= C_{\mu_B}(\mathbf{h}) \\ &= \sum_{i=1}^{N} h(x_{\pi(i)}) \left[\mu_B(S_{\pi(i)}) - \mu_B(S_{\pi(i-1)})\right] \\ &= \left( \sum_{i=1}^{k-1} h(x_{\pi(i)}) \left[\mu_B(S_{\pi(i)}) - \mu_B(S_{\pi(i-1)})\right] \right) \\ &+ h(x_{\pi(k)}) \left[\mu_B(S_{\pi(k)}) - \mu_B(S_{\pi(k-1)})\right] \\ &+ \left( \sum_{i=k+1}^{N} h(x_{\pi(i)}) \left[\mu_B(S_{\pi(i)}) - \mu_B(S_{\pi(i-1)})\right] \right). \\ &= h(x_{\pi(k)})\mu_B(S_{\pi(k)}), \end{aligned} \tag{9.5}$$

since $\mu_B(S_{\pi(i)}) - \mu_B(S_{\pi(i-1)})$ is zero except for $i = k$ and $\mu_B(S_{\pi(k-1)}) = 0$. It is trivial to show with some mathematical manipulation that the BChI in difference-in-inputs form also can be written as

$$\begin{aligned} \int_C \mathbf{h} \circ \mu_B = C_{\mu_B}(\mathbf{h}) &= \sum_{i=1}^{N} [h(x_{\pi(i)}) - h(x_{\pi(i+1)})]\mu_B(S_{\pi(i)}) \\ &= h(x_{\pi(k)})\mu_B(S_{\pi(k)}). \end{aligned} \tag{9.6}$$

According to Eqs. (9.5) and (9.6), the BChI of an instance uses only one variable $\mu_B(S_{\pi(k)})$. This fact allows us to use significantly fewer variables than the standard ChI (which we will show in Section 9.3), thus enabling the learning of a BFM for relatively larger number of inputs/sources problems, which would otherwise be intractable to solve on most personal computers.




(a) FM for $N$=4 (b) Walk for an example input

Figure 9.1
(a) FM lattice for four inputs. Arrows indicate monotonicity conditions on the immediate subsets. (b) Illustration of the path taken by observation $\mathbf{h}$ with $\mathbf{h}(\{x_2\}) \geq \mathbf{h}(\{x_1\}) \geq \mathbf{h}(\{x_4\}) \geq \mathbf{h}(\{x_3\})$. Only four variables $\mu(\{x_2\}), \mu(\{x_1, x_2\}), \mu(\{x_1, x_2, x_4\}),$ and $\mu(X)$ are used for the ChI

## 9.3 BChI learning

Let $O = \{\mathbf{h}_j, y_j\},\ j = 1, 2, \ldots, M$, be a training data set with $M$ instances. Here $\mathbf{h}_j$ represents the $j$th instance with data from $N$ inputs and $y_j$ is the associated label or ground-truth for $\mathbf{h}_j$. For example, $\mathbf{h}_j$ could be an image, $h_j(\{x_i\})$ could be the soft-max normalized decision of $N$ different learners and $y_j = 0$ if $\mathbf{h}_j$ is not the category of interest, e.g., person, or $y_j = 1$ if it is.

The goal is to learn the BFM such that the aggregation results of the training instances optimize a criteria, which is usually specified by a function of error relative to a label, $y_i$, called as an objective function. Common objective functions include the *sum of squared error* (SSE) [12, 80, 43] and the sum of absolute error (SAE). Without loss of generality, in this paper we focus only on the SSE, which is widely used due to its continuity, differentiabiliy, and non-linearity relative to errors. The sum of squared error for training data, $O$, is

$$E(O, \mathbf{u}_B) = \sum_{j=1}^{M} (C_{\mu_B}(\mathbf{h}_j) - y_j)^2,$$

where $C_{\mu_B}(\mathbf{h}_j)$ is the BChI for instance $\mathbf{h}_j$ and $y_j$ is associated label.

### 9.3.1 Learning with the full set of FM variables

Traditionally, a FM learning problem is formulated with a full set of variables without extracting any knowledge from the training data to reduce the number of variables to learn. In this case, the BChI for a training instance $\mathbf{h}_j$ is represented with a $2^N - 1$ dimensional vector $\mathbf{u}_B$, and, consequently, the SSE for training data, $O$, becomes

$$\begin{aligned} E(O, \mathbf{u}_B) &= \sum_{j=1}^{M} e^j = \sum_{j=1}^{M} (\mathbf{c}_j^T \mathbf{u}_B - y_j)^2 \\ &= ||D\mathbf{u}_B - \mathbf{y}||_2^2, \end{aligned}$$

where $D = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_M]^T$, $\mathbf{y} = [y_1\ y_2\ \ldots\ y_M]^T$, $||x||_2$ is norm-2 operation on $x$, and $\mathbf{u}_B$ is the vector of the $2^N - 1$ binary variables excluding null set. Based on this, the SSE optimization problem is

$$\min_{\mathbf{u}_B} f(\mathbf{u}_B) = ||D\mathbf{u}_B - \mathbf{y}||^2,$$

subject to

$$u_B(k) \leq u_B(l), \text{ if } u_B(k) = \mu_B(A), u_B(l) = \mu_B(B),$$

and $A \subset B, \forall k, l$ (monotonicity conditions)

$$u_B(p) = 1, \text{ if } u_B(p) = \mu_B(X) \text{ (normality conditions)}$$

$$u_B(l) \in \{0, 1\}, \forall l \text{ (BFM value restriction)},$$

which can be solved by any mixed-integer, integer, or binary quadratic programming library [147, 66]. It is obvious that this optimization problem does not scale well since the variables are exponential in $N$ regardless of the training sample size. Moreover, its complexity would be higher than the standard FM due to the use of integer-programming, which is in general costlier than the real-valued counterpart.

### 9.3.2 Efficient ChI learning

Herein, we develop an efficient algorithm that selects variables for learning from the training data instead of blindly using all FM variables in the optimization problem. As a result, the number of variables can be far fewer than the standard method, and the reductions in variables depends on different aspects of the problem at hand (e.g., training sample size, underlying FM and the problem environment–noisy or noise-free systems). In variable selection, the proposed method particularly uses the fact that the BChI computation

of an instance $\mathbf{h}_j$ requires only one variable verses $N$ variables for the real-valued FM as shown in Eq. (9.2), i.e.,

$$C_{\mu_B}(\mathbf{h}_j) = \sum_{i=1}^{N} [h_j(x_{\pi_j(i)}) - h_j(x_{\pi_j(i+1)})]\mu_B(S_{\pi_j(i)})$$

$$= h_j(x_{\pi_j(k)})\mu_B(S_{\pi_j(k)}) = h_j(x_{\pi_j(k)}),$$

where $\mu_B(S_{\pi_j(i)}) = 0, \text{ for } i < k \text{ else } 1$. In a noise-free system with underlying BFM, the the instance-error, $e_j$ in the SSE is zero, which makes the training label $y_j$ to be equal to $C_{\mu_B}(\mathbf{h}_j)$, which in turn is equal to $h_j(x_{\pi_j(k)})$. Eventually,

$$y_j = h_j(x_{\pi_j(k)}).$$

This tells us that just by inspecting the sorted input that equals $y_j$, we can determine $k$, and, ultimately, the values of variables along the walk for $\mathbf{h}_j$. However, real world problems are affected by noise (e.g., inherent or external) and/or the underlying FM may not be a 'true' BFM (here BFM will approximate the real-valued FM). In that scenario, we identify the index $k$ and associated variable $\mu^B(S_{\pi_j(k)})$ that gives the minimum instance error, $e_j$,

$$k = \arg\min_i ||y_j - h_j(x_{\pi_j(i)})||^2. \tag{9.7}$$

The noise can follow a distribution such as Gaussian, Poisson or can be purely random. Due to the variation in noise magnitude across the instances, the training instances with the same sorting order can pick different $k$'s and hence different $\mu_B(S_{\pi_j(k)})$'s, which otherwise in noise-free system would pick the same single variable.

Suppose, the instances $\mathbf{h}_l, l = \{l_1, l_2, \ldots, l_p\}$ have the same permutation, $\pi$, for their sorting order and the set of variables picked by them using Eq. (9.7) are $\mu_B(S_{\pi(k)}), k =$

$\{k_1, k_2, \ldots, k_Q\}$ with $S_{\pi(k_1)} \subset S_{\pi(k_2)} \subset \cdots \subset S_{\pi(k_Q)}$. Then the BChI for $\mathbf{h}_l$ w.r.t. these variables is

$$C_{\mu_B}(\mathbf{h}_l) = \sum_{q=1}^{Q} [h_l(x_{\pi(k_q)}) - h_l(x_{\pi(k_{q+1})})] \mu_B(S_{\pi(k_q)}), \tag{9.8}$$

where $h_l(x_{\pi(k_{Q+1})}) = 0$. Note that the number of the selected variables, $Q$ is bounded within $[1, N]$.

Let the set of variables selected by all the training instances (using Eq. (9.7)) be represented in vector form as $\mathbf{v}_B$. Then The EBChI of $\mathbf{h}_j$ can be written in matrix form as

$$C_{\mu_B}(\mathbf{h}_j) = \mathbf{a}_j^T \mathbf{v}_B,$$

where $\mathbf{a}_j$ be the coefficient of $\mathbf{v}_B$ calculated according to Eq. (9.8). Based on this, the SSE minimization problem now becomes

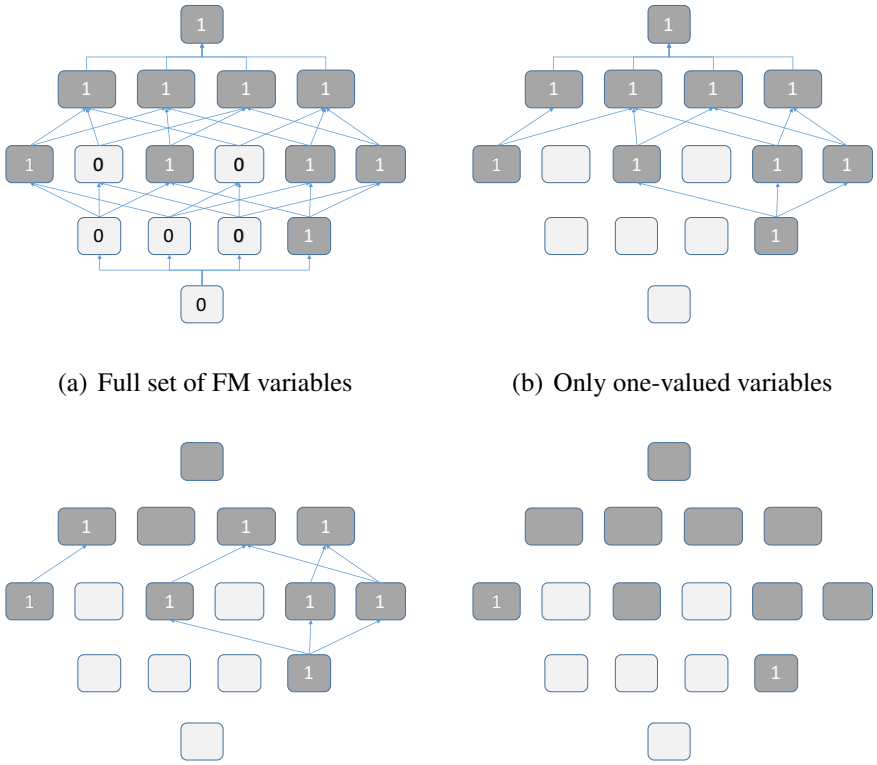
$$\min_{\mathbf{v}_B} f(\mathbf{v}_B) = ||W\mathbf{v}_B - \mathbf{y}||_2^2,$$

subject to

$$\begin{aligned}
&v_B(i) \leq v_B(j), \text{ if } v_B(i) = \mu(A), v_B(j) = \mu(B) \\
&\text{and } A \subset B, \forall i, j \in \{1, 2, \ldots, Q\} \text{ (monotonicity conditions)} \\
&v_B(j) = 1, \text{ if } v_B(j) = \mu_B(X) \text{ (normality conditions)} \\
&v_B(i) \in \{0, 1\}, \ \forall i \text{ (BFM value restriction)}
\end{aligned} \tag{9.9}$$

where

$$W = [\mathbf{a}_1 \ \mathbf{a}_2 \ \ldots \ \mathbf{a}_M]^T.$$



(a) Full set of FM variables

(b) Only one-valued variables

(c) Training variables from an arbitrary data set

(d) Independent variables

Figure 9.2
An example of the BFM representation for four inputs case. Light gray nodes with zeros represent zero-valued variables while dark-grey nodes with one's denote one-valued variables. Empty nodes are for placeholders only, and indicate that their variables are removed. (a) shows the full set of FM variables, (b) only one-valued variables, (c) EBChI variables selected from a noise free training data set, and (d) EBFM represented with independent variables. It can be seen from (d) that there is no partial order or monotonicity conditions defined among independent variables. The full FM lattice (a) can be simply derived from (d) using the FM's monotonicity property.

It is to be noted that instead of selecting one variable per instance as in Eq. (9.7), we can use different criteria to select multiple variables per instance, e.g., selecting variables associated with those inputs that fall within a certain threshold (or standard deviation) of the training label $y_j$. While increasing the number of variables will have no impact for a system with little noise with sufficient training samples, it can lower the SSE for a highly

noisy system with limited data. However, increasing variables gives diminishing results, so for a given problem there is a trade-off between the number of variables and the error of the objective function.

## 9.4 Efficient BFM data structure

In the last section, we provided an algorithm to efficiently learn a BFM that uses fewer variables. Anderson et al. introduced a simple approach to represent the BFM that can be applied here to further reduce the learned variables for efficient storage and representation. In that method, the variable elimination process considers only the values of the variables and does not take into account the properties of the FM, which can greatly enhance the representation technique. By taking into consideration both values and properties, herein we propose a new way to efficiently represent the full-fledged BFM and then we provide an upper bound on the minimum number of variables required.

### 9.4.1 Representation

Since the variables of a BFM are binary valued, the variables can take either zero or one values. The zero-valued variables do not contribute to the BChI, so they can be discarded. Thus, only one-valued variables can be considered as candidates for representation. Due to the FM's monotonicity property, if a variable $\mu_B(A)$ is one-valued, then all variables that are for the supersets of $A$ are also one-valued. As such, the one-valued variables can be divided into two parts: (i) independent variables whose values cannot be derived from another one-valued variable using monotonicity condition and (ii) dependent variables whose values can be retrieved using the independent variables and monotonicity condition, and

therefore, can be discarded. Consequently, only the one-valued independent variables are necessary to represent a BFM, which we refer to as EBFM. Figure 9.2 illustrates the EBFM representation technique for an example with $N = 4$.

From the EBFM with independent variables that correspond to sets $B = \{B_1, B_2, \ldots, B_l\}$, the BChI of an observation $\mathbf{h}_j$ can be computed as follows:

1. Sort inputs in descending order. Let the sorting order be $x_{\pi_j(i)}, i = 1, 2, \ldots, N$, and the associated variables for the BChI are $\mu_B(S_{\pi_j(i)}), i = 1, 2, \ldots, N$.

2. Find the minimum $k$ for which $S_{\pi_j(k)} \supseteq B_l \in B, \forall B_l$.

3. Return $\mathbf{h}_j(x_{\pi_j(k)})$ as the output.

### 9.4.2 Upper bound

To determine the upper bound on the number of variables in EBFM, we use a theorem by E. Sperner [172]. The theorem proves that if $B_1, B_2, \ldots, B_t$ are subsets of an $N$-element set $B$, such that no $B_i$ is a subset of any other $B_j$, then

$$t \leq \binom{N}{[N/2]}, \tag{9.10}$$

where $[x]$ denotes the rounded integer value. As the independent variables in EBFM have the same definitions as the $B_i$s above, Eq. (9.10) also gives the upper bound on the number of variables in the EBFM. For $20$ inputs, there can be at most $184,756$ independent variables (in median-like aggregation case), thus using only $17.62\%$ of the total variables in the worst case. However, the actual usage (or saving) depends on the specific BFM; for example, min and max aggregation operators require $1$ and $N$ variables respectively.

## 9.5 Experiments

Experiments are conducted on synthetic data set for the following reasons. First and foremost, we know the true underlying BFM, which facilitates the comparison and investigation of the proposed method's behaviour, whereas in real world applications/data the true FM may never be known. Moreover, it is quite challenging to find real world examples of varying complexity while it is far easier to create a FM in synthetic data with different complexity. Synthetic experiments also give us insight into how the learning method will behave in different noisy contexts. The experiments are designed to compare the computational complexity as well as to measure the performance in terms of *mean squared error* (MSE) from the predicted test labels using the learned BFM in a no noise as well as in a noisy environment.

### 9.5.1 No noise scenario

First, a training data set of $M = 500$ and $N = 8$ is generated pseudo-randomly from a uniform distribution, which is then partitioned to create five data-sets for five fold cross-validation. Each cross-validation data-set contains $400$ training samples and $100$ test samples. Then we created training data-sets of sample sizes $150$, $75$, $30$, and $15$ via random selection of instances from those of $400$, $150$, $75$, and $30$ respectively. Test data for all sample sizes remains the same. We specified three BFMs–BFM1, BFM2 and BFM3–in Table 9.1 using the EBFM representation with independent variables. In the table, each independent variable is denoted with the inputs' indices in the set, e.g., $12$ stands for independent variable $\mu_B(\{x_1, x_2\})$. The independent variables in BFM1 lie in the lower part

of the lattice (hence largest number of one valued variables) while those of BFM3 reside on the upper part. The BFM2 independent variables spread across the lattice from top to bottom. The labels for these BFMs are created without adding any noise, which also serve as the ground-truth for noisy system.

Table 9.1
The FMs used in the experiment

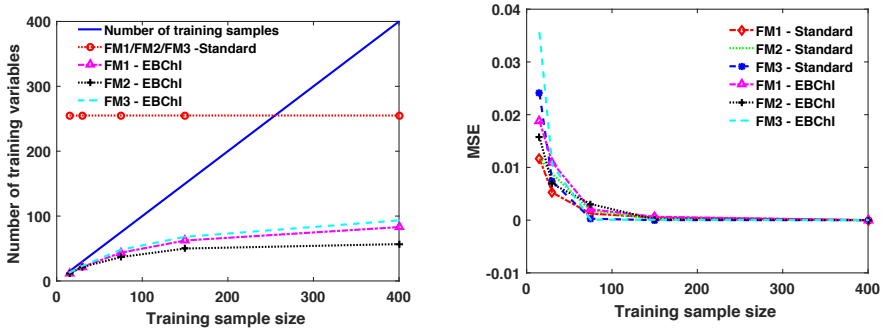
| BFM | Independent variables | | | | | | | | | | | | | | #var | # 1-var |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 56 | 67 | 57 | 345 | 346 | 347 | 1234 | 1235 | 1245 | 1236 | 1246 | 1237 | 1247 | 255 | 211 |
| 2 | 3568 | 3578 | 3678 | 4568 | 4578 | 4678 | 5678 | 12568 | 12578 | 12678 | 123458 | 123468 | 123478 | 1234567 | 255 | 59 |
| 3 | 67 | 68 | 78 | 123456 | 123457 | 123458 | | | | | | | | | 255 | 131 |

Figure 9.3 shows the results for different sample sizes. As can be seen, the number of variables increases linearly at small $M$, then remains constant for large sample sizes, which is still far fewer than the standard optimization method. The average of the MSE as well as the number of variables correctly learned are approximately the same for both standard and EBChI (Figure 9.3(b) and (c)). An interesting observation from Figure 9.3(d) is that the EBChI has far fewer independent variables for $15$ training samples, meaning the learned FM from the EBChI is less complex than the standard one.

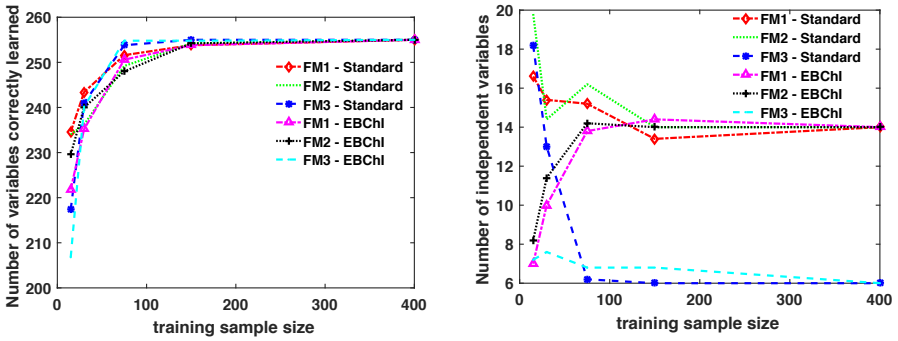
### 9.5.2 Noisy scenario

In most analysis of the noisy systems, noise is usually modeled as Gaussian distribution, which provides a very good approximation in many real world scenarios. Herein, we model the output as $y = C_{\mu_B}(\mathbf{h}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The observations in the training data set are the same as those for the noise-free system; however, the labels are created

by adding randomly generated values from a normal distribution of variance $\sigma_n^2$. We conducted experiments with five different variances, $\sigma_n/\sigma_y = \{0, 0.01, 0.05, 0.1, 0.3, 0.5\}$, where $\sigma_y^2$ is the variance of the true labels. The standard deviations for FM1, FM2, and FM3 are 0.184, 0.1804, and 0.2055 respectively. We measured the MSE with respect to the true test labels.




(a) Average number of training variables (b) Average mean squared error on test data over five iterations




(c) Average number of variables correctly learned (d) Average number of independent variables

Figure 9.3

Results for data-driven learning from noise-free training data of different sample sizes, $M = 15,\ 30,\ 75,\ 150,$ and $400$.

Figure 9.4 compares the results for noisy data with $400$ training samples. More noise means more variations around the true value, which results in selection of multiple vari-

ables for instances with the same sorting order. Thus, the number of training variables in EBChI increases with the noise level. The presence of noise equally affects both the EBChI and the standard BChI. When the FMs are learned with sufficient number of samples, the error is minimal–on the order of $10^{-}3$ (Figure 9.4(c))–and there is a mismatch of only $4$ out of $255$ variables in the worst scenario; see Figure 9.4(c). As the result shows, both the standard method and EBChI are resilient to relatively moderate level of noise ($\sigma_n/\sigma_y = 0.3$).

## 9.6 Conclusion

In this paper, we proposed an efficient method to learn the BFM. Variable selection in the EBChI is driven by the observed instances whereas the standard learning method uses full set of variables. This makes the EBChI tractable for a relatively large problem (large $N$) in contrast to the standard approach. As demonstrated by the results, learning with the EBChI is approximately equivalent to the standard BChI learning method for noisy and noise-free scenarios; therefore, it provides an efficient alternative for data-driven learning of the BFM. Moreover, we introduced a representation technique called the EBFM to describe a BFM minimally via independent variables. In future work, we will apply our technique to real world problems. Additionally, we will study which problems can be natural fit to BFM and which problems can be approximated by a BFM.

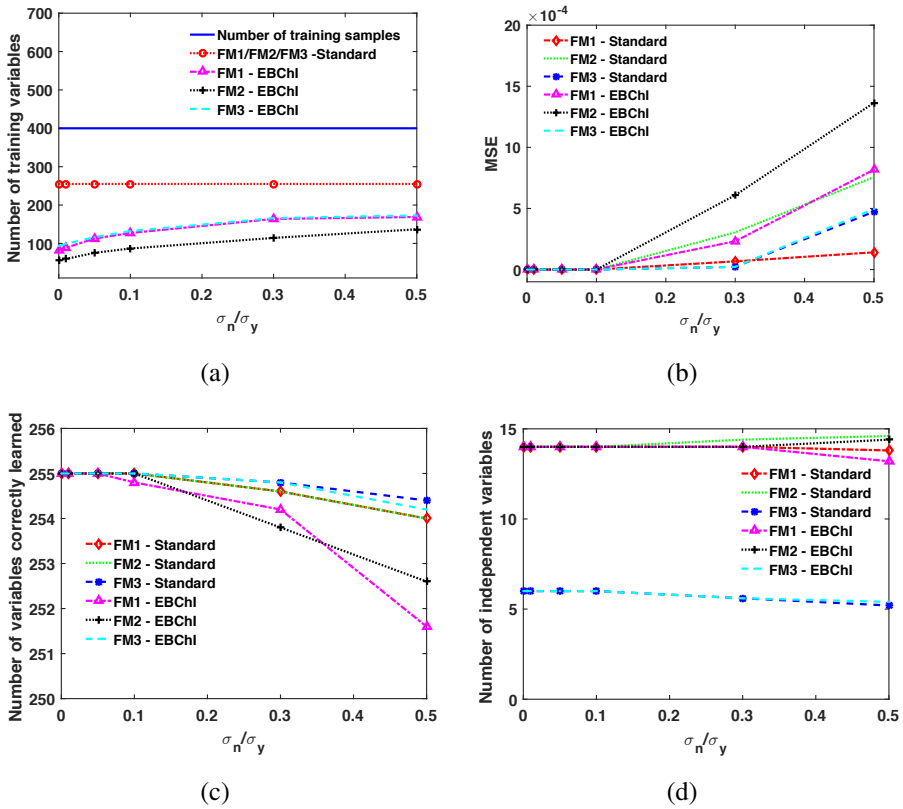



Figure 9.4

Results for data-driven learning with noise, with standard deviation, $\sigma_n = \{0.0\sigma_y, \}$ . (a) average number of training variables over five iterations, (b) average mean squared error on test data, (c) average number of variables correctly learned, (d) corresponding independent variables

# CHAPTER X

# EXPLAINABLE AI FOR UNDERSTANDING DECISIONS AND DATA-DRIVEN OPTIMIZATION OF THE CHOQUET INTEGRAL

## 10.1 Introduction

Data and information fusion is a fundamental capability in many state-of-the-art technologies, e.g., Big Data, smart cars, remote sensing, computer vision, etc. However, fusion is a rather vague concept that takes many forms. For example, in remote sensing sensor registration is fusion. In computer vision, the combining of features is fusion. In expert systems, the combining of expert opinions is fusion. Herein, we restrict our analysis and focus on a more specific avenue of fusion: aggregation functions. Let $X = \{x_1, ..., x_N\}$ be $N$ sources like a sensor, human or algorithm. In general, an aggregation function is a mapping of data from our $N$ sources, denoted by $h(x_i) \in \Re$, to data, $f(\{h(x_1), ..., h(x_N)\}, \Theta) \in \Re$, where $\Theta$ are the parameters of $f$. Fusion typically needs a "home" to make sense, e.g., fusion for machine learning, fusion for scientific visualization, etc. The point is, in order to evaluate the success of fusion one typically needs a context in which to explore the quality of its result. Common nomenclature for fusion of different data is *signal-in-signal-out* (SISO), *feature-in-feature-out* (FIFO) and *decision-in-decision-out* (DIDO). Herein, we focus on Sugeno's fuzzy *Choquet integral* (ChI), which has been used for SISO, FIFO and DIDO.

Often, it is too complicated or not possible for a human to specify the parameters of the underlying aggregation operator. In response, a multitude of methods have appeared on data-driven optimization of aggregation operators, e.g.,[111]. The subject of this article is, learning a solution from data is useful, but what is the quality of that learned solution and why should we trust decisions that it produces on new data? In part, the current article is driven by the emerging need for so-called *explainable artificial intelligence* (XAI). Herein, we explore XAI methods for introspection of the ChI. As such, we pull together existing methods and explore new data-driven ones.

In Section 10.2 we review the ChI, Section 10.3 is existing *fuzzy measure* (FM) specific XAI tools, Section 10.4 is existing ChI specific XAI methods, and 10.5 is new data-centric XAI methods. Last, in Section 10.6 we use these XAI methods to fuse a heterogeneous set of *deep convolutional neural networks* (DCNN) for remote sensing on benchmark data sets.

## 10.2 Measure and Choquet Integral

The fuzzy integral has been demonstrated numerous times in a variety of applications; e.g., explosive hazard detection [167, 62], computer vision [181], pattern recognition [77, 137, 118], multi-criteria decision making [70, 127], forensic anthropology [6, 14], fuzzy logic [187], multiple kernel learning [152], multiple instance learning [55], ontologies [3], missing data [110], and deep learning for remote sensing [5, 164], to name a few. The ChI is a nonlinear aggregation function parameterized by the FM. Herein, we focus on the real-valued discrete (finite $X$) ChI.

The first action we face is how to assign "worth" to different subsets of sources. For example, the well-known backbone of calculus on real-valued domains is the Lebesgue measure; which coincides with length, area and hypervolume. However, when $X$ is a discrete domain, e.g., set of algorithms, what is the corresponding measure? In [181], Keller et al. first investigated the idea of using the fuzzy integral for pattern recognition. A FM is a function, $\mu$, on the power set of $X$, $2^X$, which satisfies (1) (boundary condition) $\mu(\emptyset) = 0$ and (2) (monotonicity) if $A, B \subseteq X$ and $A \subseteq B$, then $\mu(A) \leq \mu(B)$. Often, $\mu(X) = 1$ is imposed in settings like DIDO fusion.

The FM models *interactions* (e.g., subjective worth, statistical correlation, etc.) between input subsets. The data provided by our inputs are $\{h(\{x_1\}), h(\{x_2\}), ..., h(\{x_N\})\}$. The fuzzy integral is a way to combine the integrand ($h$) data relative to the FM ($\mu$). Let $h(\{x_i\}) \in \Re^{\geq 0}$ be the data/information from input $i$. The discrete (finite $X$) Sugeno FI is[1]

$$\int_S h \circ \mu = S_\mu(h) = \bigvee_{i=1}^{N} \left( h(\{x_{\pi(i)}\}) \wedge \mu(A_i) \right), \tag{10.1}$$

where $\pi$ is $h(\{x_{\pi(1)}\}) \geq h(\{x_{\pi(2)}\}), \ldots, \geq h(\{x_{\pi(N)}\})$ and $A_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$. The discrete ChI is[2,3]

$$\int_C h \circ \mu = C_\mu(h) = \sum_{i=1}^{N} h(\{x_{\pi(i)}\}) \left[ \mu(A_i) - \mu(A_{i-1}) \right], \tag{10.2}$$

[1]Due to the maximum (t-conorm) and minimum (t-norm) operators, the Sugeno FI does not actually generate any possible number between the minimum and maximum of the inputs. Instead, it selects one of the FM or input values, i.e., at most one of $2^N + N$ values.

[2]The ChI is used frequently for a number of reasons; e.g., it is differentiable [137], for an additive (probability) measure it recovers the Lebesgue integral, it yields a wider (versus the Sugeno integral) spectrum of values, etc.

[3]If $\mu(X) < 1$, idempotence, boundedness, etc. are not guaranteed.

where $\mu(A_0) = 0$. Since the ChI is a parametric aggregation function, once the FM is determined the ChI turns into a specific operator. For example: if $\mu(A) = 1, \forall A \in 2^X \setminus \emptyset$, the ChI becomes the maximum operator; if $\mu(A) = 0, \forall A \in 2^X \setminus X$, we recover the minimum; if $\mu(A) = \frac{|A|}{N}$, we recover the mean; and for $\mu(A) = \mu(B)$ when $|A| = |B|$, $\forall A, B \subseteq X$, we obtain a *linear combination of order statistics* (LCOS). In general, each of these cases can be viewed as constraints or simplifications on the FM (and therefore the ChI).

The discrete ChI can alternatively be regarded as $N!$ *linear convex sum* (LCS) operators–one for each possible input sort.[4] For a single *instance* of the ChI, i.e., $N$ new inputs that need to be fused, its sort determines which LCS to use. If we consider a constrained FM then this set of LCSs is reduced. For example, a LCOS has one underlying operator. Each sort has a corresponding "walk" in the FM. For example, let $N = 3$ and $h(\{x_2\}) \geq h(\{x_3\}) \geq h(\{x_1\})$. Equation (10.2) thus uses the FM variables $\mu(\emptyset)$, $\mu(\{x_2\})$, $\mu(\{x_2, x_3\})$ and $\mu(X)$. This sequence of increasing size (cardinality) FM variables is referred to as a "walk" up the FM lattice hereafter[5].

### 10.2.1 Data-Driven Optimization of the ChI

A big question is, where do we get the FM from? One option is to have an expert specify it. However, this is not practical (assuming the expert could even meaningfully assign values to the interactions) as the number of inputs increases. Another option is we

[4]Whereas there are $N!$ LCS operators, they share $2^N$ variables. For example, for $N = 6$ we have 720 LCS operators but only 64 variables.

[5]As $\mu(\emptyset) = 0$ and $\mu(X) = 1$, it is trivial to prove that the difference-in-measure coefficients in Equation (10.2) sum to 1.

can specify or learn the worth of just the singletons (the densities). From there, a number of formulas can be used to impute (fill in) the missing variable values. Popular approaches include the Sugeno $\lambda$-FM and the S-Decomposable FM [176]. However, while convenient, most often we do not obtain the desired values for variables that we need. Next, we quickly review one method of optimizing the ChI. The reader can refer to [111] for more details (full mathematical detail and experiments).

Let $O = \{\mathbf{h}_j, y_j\},\ j = 1, \ldots, M$, be $M$ training examples; where $\mathbf{h}_j$ is the $j$-th instance with data/information from $N$ inputs and $y_j$ is the ground-truth for $\mathbf{h}_j$. The sum of squared error for training dataset $O$ is

$$E(O, \mathbf{u}) = \sum_{j=1}^{M} e^j = \sum_{j=1}^{M} (\mathbf{c}_j^T \mathbf{u} - y_j)^2 = ||D\mathbf{u} - \mathbf{y}||_2^2, \tag{10.3}$$

where $\mathbf{u} = [\mu(\{x_1\}), ..., \mu(\{x_1, x_2\}), \mu(\{x_1, x_3\}), ..., \mu(X)]$ (lexiographic vector of size $2^N - 1$), $D = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_M]^T$ (full dataset), $\mathbf{y} = [y_1\ y_2\ \ldots\ y_M]^T$, $||\cdot||_2$ is norm-2 operation, and $\mathbf{c}_j$ holds the coefficients of $\mathbf{u}$ for observation $\mathbf{h}_j$, e.g., for $N = 3$ and $h(\{x_2\}) \geq h(\{x_1\}) \geq h(\{x_3\})$, $c$ is

$$[0, h(\{x_2\}) - h(\{x_1\}), 0, h(\{x_1\}) - h(\{x_3\}), 0, 0, h(\{x_3\})]\,.$$

The regularized SSE optimization problem is

$$\min_{\mathbf{u}} f(\mathbf{u}) = ||D\mathbf{u} - \mathbf{y}||^2 + \beta v(\mathbf{u}), \tag{10.4}$$

where $\beta \in \Re^{\geq 0}$ is a regularization constant (which balances the "cost" (or penalty) of obtaining minimum function error relative to our desire to have minimal model complexity) and $v(\mathbf{u})$ is an index of model complexity (e.g., k-additive and Mobius, Gini-Simpson, $\ell_p$-norm, etc. [149]), subject to the FM boundary and monotonicity conditions (see [12] for

how to pack the constraints into a linear algebra expression), which can be solved via quadratic programming. Full code and explanation (including how to build the constraint matrix $C$) can be found at `www.derektanderson.com/FuzzyLibrary`.

### 10.2.2 Data Supported and Unsupported Variables

Many parametric methods are unable to determine which variables are supported (and to what degree at that) by data. For example, determination of which parameters are sufficiently approximated by data in a neural network is unsolved. In [111], we put forth a way to identify data supported and data unsupported FM variables for learning the FM/ChI from data. For $M$ training instances we get, at maximum, $M$ unique sorts[6]. For each walk, we record which variables are used. Data supported means a FM variable ($A \subseteq X$) was encountered at least once. Data unsupported means a variable was never encountered in the $M$ walks.

Figure 10.1 shows the $2^N$ FM variables for an $N = 4$ problem–specifically the fusion of DCNNs. "Layer" $L$ (from bottom to top) in the image denotes FM variables with cardinally $L$. Thus, layer $0$ (bottom node) is the empty set, the next layer is the singletons (from left to right), top is $\mu(X)$, etc. Each variable is presented in lexicographic order, i.e., layer $2$ is $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_1, x_4\}$, $\{x_2, x_3\}$, $\{x_2, x_4\}$ and $\{x_3, x_4\}$. The nodes are also drawn size-wise proportional to their value (a minimum size and maximum was specified to make them still show up for $0$-valued variables). In addition, the "paths" drawn indicate the visitation frequency (the brighter the line, the higher the visitation) for test data in fold 1.

[6]In practice, it is not common that we encounter $M$ unique sorts for $M$ instances. Depending on the diversity of our data, it is more common to have just a small percentage of unique walks. The number of walks typically becomes sparser as $N$ increases.

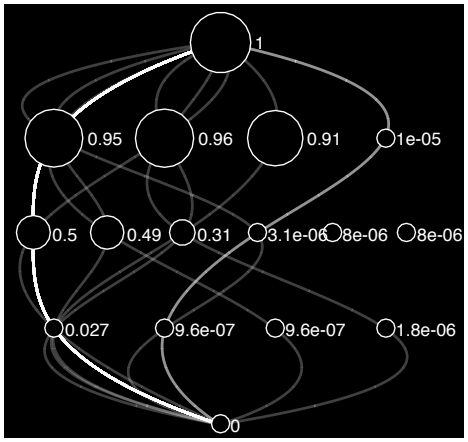



Figure 10.1
Visualization of the FM and walk visitation frequency for the fusion of four deep convolutional neural networks for neuron one (agricultural) in the remote sensing UCM dataset. The most frequently encountered walk is $\{x_1\}$, $\{x_1, x_2\}$, $\{x_1, x_2, x_3\}$, then $\{x_1, x_2, x_3, x_4\}$ (which corresponds to the DCNNs CaffeeNet, GoogleNet, ResNet50 and ResNet101).

## 10.3 Measure-Centric Indices: $f_{\mathbf{mc}}(\mu)$

Next, we review the first of three approaches to understand the inner-workings of fusion. In general, we have $(\mu, C_\mu, O)$ at our disposal–the measure, integral and data set respectively. This section focuses on questions that require just $\mu$.

### 10.3.1 Shapley Index

The Shapley index informs us about the "worth" of each input,

$$\Phi_\mu(i) = \sum_{K \subseteq X \backslash \{i\}} \zeta_{X,1}(K) \left(\mu(K \cup \{i\}) - \mu(K)\right), \tag{10.5}$$
$$\zeta_{X,1}(K) = \frac{(|X| - |K| - 1)!|K|!}{|X|!},$$

where $K \subseteq X \backslash \{i\}$ denotes all proper subsets from $X$ that do not include source $i$. The Shapley value of $\mu$ is the vector $\Phi_\mu = (\Phi_\mu(1), ..., \Phi_\mu(N))^t$ and $\sum_{i=1}^{N} \Phi_\mu(i) = 1$. The

Shapley index can be interpreted as the average amount of *contribution* of source $i$ across all coalitions.

### 10.3.2 Interaction Index

The Interaction Index informs us about how two inputs *interact* with one another–aka what advantage (or not) is there in combining inputs. The Interaction Index (Murofushi and Soneda [142]) between $i$ and $j$ is

$$\begin{aligned} \mathcal{I}_\mu(i,j) = & \sum_{K \subseteq X \setminus \{i,j\}} \zeta_{X,2}(K)(\mu(K \cup \{i,j\}) \\ & - \mu(K \cup \{i\}) - \mu(K \cup \{j\}) + \mu(K)), \qquad (10.6) \\ \zeta_{X,2}(K) = & \frac{(|X| - |K| - 2)!|K|!}{(|X| - 1)!}, \end{aligned}$$

where $\mathcal{I}_\mu(i,j) \in [-1,1], \forall i,j \in \{1,2,...,N\}$. A value of 1 (respectively, $-1$) represents the maximum complementary (respective redundancy) between $i$ and $j$. The reader can refer to [78] for further details about the interaction index, its connections to game theory and interpretations. Grabisch extended the index to the general case of any coalition [74],

$$\begin{aligned} \mathcal{I}_\mu(A) = & \sum_{K \subseteq X \setminus A} \zeta_{X,3}(K,A) \sum_{C \subseteq A} (-1)^{|A \setminus C|} \mu(C \cup K), \qquad (10.7) \\ \zeta_{X,3}(K,A) = & \frac{(|X| - |K| - |A|)!|K|!}{(|X| - |A| + 1)!}. \end{aligned}$$

Equation (10.7) is a generalization of both the Shapley index and Murofushi and Soneda's interaction index as $\Phi_\mu(i)$ corresponds with $\mathcal{I}_\mu(\{i\})$ and $\mathcal{I}_\mu(i,j)$ with $\mathcal{I}_\mu(\{i,j\})$.

## 10.4 Integral-Centric Indices: $f_{\mathbf{ic}}(\mu, C_\mu)$

A different fundamental question is what "type" of aggregation is the ChI performing? Answering this question helps us understand how the inputs are being combined (e.g., in an optimistic, pessimistic, expected value like fashion, etc.). In this section we review indices that operate on $(\mu, C_\mu)$.

In [154], we established an index $D_1$ to measure the degree to which a given FM/ChI is an maximum operator. Let "layer k" (measure defined on sets of cardinality $k$) be denoted by $L(k)$, e.g., $L(1) = \{\mu(\{x_1\}), \mu(\{x_2\}), \mu(\{x_3\})\}$ for $N = 3$.

$$D_1(\mu) = \sum_{k=1}^{1} \frac{\mathbf{W}(k)}{2}(T_1 + T_4) + \left[\sum_{k=2}^{N} \frac{\mathbf{W}(k)}{3}(T_1 + T_2 + T_4)\right], \tag{10.8}$$

$$T_1 = 1 - \left(\frac{\sum_{I \in L(k)} \mu(I)}{|L(k)|}\right),$$

$$T_2 = \left(\frac{\sum_{I \in L(i)} \mu(I)}{|L(k)|} - \frac{\sum_{J \in L(k-1)} \mu(J)}{|L(k-1)|}\right),$$

$$T_3 = \frac{\sum_{I \in L(k)} \mu(I)}{|L(k)|},$$

$$T_4 = \frac{\sum_{I \in L(k)} (\mu(I) - T_3)^2}{|L(k)| - 1},$$

$$\mathbf{W} = \frac{[\frac{1}{N}, ..., 1]}{\sum_{i=1}^{N} \frac{i}{N}}.$$

In summary, $T_4$ is the variance of a layer, $T_3$ is the mean of a layer, $T_2$ is the difference in mean value between two consecutive layers, $T_1$ is how far the mean of a layer is from value 1 and $\mathbf{W}$ is a set of layer weights. A value of $D_1 = 0$ means that a ChI is the maximum

operator. In addition, the distance of a learned capacity to a minimum operator ($D_2$), mean ($D_3$) and LCOS ($D_4$) is

$$\begin{aligned} D_2(\mu) &= \sum_{k=1}^{1} \frac{\mathbf{W}_2(k)}{2} \left(T_3 + T_4\right) \\ &+ \left[\sum_{k=2}^{N-1} \frac{\mathbf{W}_2(i)}{3} \left(T_3 + T_2 + T_4\right)\right], \end{aligned} \tag{10.9}$$

$$D_3(\mu) = \frac{1}{2^N - 2} \sum_{k=1}^{N-1} \sum_{I \in L(k)} \left| \mu(I) - \frac{k}{N} \right|, \tag{10.10}$$

$$D_4(\mu) = \frac{1}{N-1} \sum_{k=1}^{N-1} \sqrt{T_4}, \tag{10.11}$$

$$\mathbf{W}_2 = \frac{\left[1, ..., \frac{1}{N-1}\right]}{\sum_{i=1}^{N-1} \frac{i}{N-1}}.$$

The formulas in this section provide a way to measure the divergence of a learned FM to a reference FM (and thus underlying ChI). However, the caveat is these indices are distance measures. As such, their values are not directly membership degrees. If one desires such information then a membership function needs to be specified or learned.

## 10.5 Data-Centric Indices: $f_{\mathbf{dc}}(\mu, C_\mu, O)$

In this section, we create new indices for answering data-driven questions. Ultimately, these results help us understand the diversity (or lack of) contained in $O$. Last, we explore a way to determine how much we should trust the result of fusion for data not in our training set. Meaning, have we seen its like before and how confident should we be in the result coming out of our system (fusion operator)?

### 10.5.1 Variable Visitation

In this subsection we address the question of *how frequently is a FM variable encountered in training data?* Algorithm 9 is a formal description of the procedure.

**Algorithm 3:** Variable Visitation

INPUT: Data set $O$ ▷ *$M$ instances with $N$ inputs each*

INPUT: FM $\mu$ ▷ *Our fuzzy measure*

Initialize each variable in $\mathbf{v}$ (where $|\mathbf{v}| = |\mathbf{u}|$) to zero

**for** $j = 1$ *to* $M$ ▷ Evaluate each instance **do**

    For $\mathbf{h}_j$, identify the $N - 1$ FM variables (excluding the empty set and $X$) used in Equation (10.2). Let these $N - 1$ indices be denoted as $\mathcal{I}_j = \{I_{j,1}, ..., I_{j,N-1}\}$, where $I_{j,1}$ is the index for a density, $I_{j,3}$ is for a three tuple, etc.

    **for** $k = 1$ *to* $N - 1$ **do**

        Set $\mathbf{v}(I_{j,k}) = \mathbf{v}(I_{j,k}) + 1$.

Set $\mathbf{v} = \frac{\mathbf{v}}{M}$.

RETURN: $\mathbf{v}$

Algorithm 9 counts how many times a FM variable is encountered in the $M$ different ChI instances. It then normalizes these counts to obtain their relative frequency of occurrence per layer. Each layer has a different number of variables. Consider the case of $N = 4$. Layer one (densities) has four variables and layer two (tuples) has six variables. Assume there are four training samples, $M = 4$, so four walks. It is possible for each walk to touch a different density. However, these four walks can only touch four of the six tuples. We normalize per layer to provide an answer that supports *what is the relative frequency of occurrence of variables at each layer*, i.e., how often did I encounter this three

tuple relative to other three tuples? Last, data supported variables are derived by looking at entries in $\mathbf{v}$ whose value is greater than zero. Data unsupported are entries in $\mathbf{v}$ with value (probability) zero.

### 10.5.2 Percentage of Data Supported Variables

The next question we address is *overall, what percentage of our FM variables are supported by data?* To answer this question we first run Algorithm 9. Next, we "harden" $\mathbf{v}$—meaning we convert $\mathbf{v}$ into $\hat{\mathbf{v}}$ such that an index in $\hat{\mathbf{v}}$ is $0$ if its corresponding value in $\mathbf{v}$ is $0$, otherwise it is assigned a $1$. Next, we calculate an auxiliary variable,

$$\delta_1 = \sum_{k=1}^{|\hat{\mathbf{v}}|} \hat{\mathbf{v}}(k).$$

The answer to our question is

$$i_1(\mu, O) = \frac{\delta_1}{|\hat{\mathbf{v}}|} \in [0, 1]. \tag{10.12}$$

### 10.5.3 Walk Visitation

The last two subsections focused on individual variables. The next index (see Algorithm 3) we explore is based on the fact that the discrete (finite $X$) ChI is in reality $N!$ LCS operators. As such, we need to know which of these operators are supported by training data.

Algorithm 3 simply iterates each instance in our training data, determines its sort, and a mapping function (from sort to index in the FM) is used to increment the counts. At the end, these counts are normalized by the number of instances ($M$).

[8]For example, let $N = 2$ and $h_1(\{x_2\}) \geq h_1(\{x_1\})$. The sort index (walk) would therefore be $[2, 1]$.
[8]Note, $r(\cdot) \in \{1, ..., N!\}$ is the index resolving function for sort $\mathbf{s}_j$.

**Algorithm 4:** Walk Visitation Frequency

1 INPUT: Data set $O \triangleright M$ *instances with* $N$ *inputs each* INPUT: FM $\mu \triangleright$ *Our fuzzy measure* Let $\mathbf{z}$ (size $N!$) be a vector of all zeros **for** $j = 1$ *to* $M$ ▷ Evaluate each instance **do**

2 Sort (in decending order) the values in $\mathbf{h}_j$. The result is $\mathbf{s}_j = [s_{j,1}, ..., s_{j,N}]$.[7]Let each sort map to a unique index in $\mathbf{z}$. As such, let $\mathbf{z}(r(\mathbf{s}_j)) = \mathbf{z}(r(\mathbf{s}_j)) + 1$.[8]

3 Divide each variable in $\mathbf{z}$ by $M$. RETURN: $\mathbf{z}$

### 10.5.4 Percentage of LCSs Observed

Algorithm 3 determines the relative frequency of occurrence for each walk. The next index is a summarizing statistic of *what percentage of walks were encountered?* This is calculated in a similar respect to index $i_1$. First, we convert $\mathbf{z}$ into a vector of zeros and ones ($\hat{\mathbf{z}}$). The index is then calculated as $i_2(\mu, O) = \frac{\delta_2}{N!} \in [0, 1]$, where $\delta_2$ is the sum of the $\hat{\mathbf{z}}$ values.

### 10.5.5 Dominant Walk Identification

The next index asks the high-level summarizing question of, *is our training data diverse, or does there exist a dominant walk?* This index, $i_3(\mu, O)$, is

$$i_3(\mu, O) = \max_k \mathbf{z}(k). \tag{10.13}$$

If the training data has ideal walk variety then $\mathbf{z}$ is a uniform distribution. On the other hand, when diversity is poor we drive towards a single value of one and zero elsewhere. Thus, higher values for $i_3(\mu, O)$ mean less diversity in data and therefore our fusion solution may suffer.

### 10.5.6 Should we Trust our Fused Result?

The above techniques are focused on answering questions about variable and walk visitation and data diversity. However, at the end of the day we are typically given a set of data $O$, that may or may not be diverse, and we have to learn from what we are given. An important question is, *if we are given a new instance ($N$ inputs)–e.g., a sample from test data–can we trust the fused result?* Whereas this might seem like a simple question, it is an important and fundamental one. Algorithm 7 is our proposed method.

**Algorithm 5:** Should We *Trust* the Fused Result?

INPUT: new instance $\mathbf{h}$
INPUT: learned $\mu$
Set $t = 0$
**for** $k = 1$ *to* $N - 1$ **do**
    Obtain the walk for $\mathbf{h}$.
    For each variable encountered in the walk, if its corresponding variable value in $\mathbf{v}$ is $0$, increment $t$.
Set $t = 1 - \frac{t}{N-1}$. RETURN: $t$

Algorithm 7 yields a value $t \in [0, 1]$. This value is one when we have encountered data in training to support all of its relevant FM variables. However, as $t$ decreases, we are relying on variables (parts of our fusion calculation) that were not supported by data but are most likely a result of the optimization process. As we showed in [111], this often depends on the optimizer. For example, in simple quadratic programming data unsupported variables will take the lowest possible value to ensure monotonicity. Is this the "correct"

value that should have been used? Most likely, no. The point is, a system can now return the tuple $(C_\mu(\mathbf{h}), t)$, i.e., what is the result of fusion and how much might we trust this in terms of what percentage of the answer was supported by data.

### 10.6 Experiments and Results

Herein, we fuse four heterogeneous architecture DCNNs–CaffeeNet, GoogleNet, ResNet50 and ResNet101–on the *UC Merced* (UCM) remote sensing dataset. The UCM includes 21 classes that are a mix of objects and landcover (see Table 10.1). Each image chip is 256x256 in spatial size and each pixel is approximately 0.3m *ground sampling distance* (GSD) spatial resolution. Some classes, e.g., harbor and parking lot, are complex compositions of sub-entities (boats and vehicles); while others are general structural patterns of shapes (e.g., intersection and baseball diamonds). In general, the variability and complexity of overhead imagery is immense as visual cues exist at multiple levels: fine-scale (e.g. airplane shapes, vehicle presence, etc.) to large-scale (e.g., road way configurations in overpasses versus intersections versus freeway).

The DCNNs were trained using our procedure outlined in [163], which includes transfer learning and data augmentation. The trained DCNNs are then used in a *locked state*, i.e., no further learning happens in DL during the fusion stage. The training of the DCNNs are done in five-fold, cross validation manner; such that we have 5 sets of 80% training and 20% testing for both datasets. Per DCNN fold, three-fold CV fusion is used. An approximately equal number of samples are randomly selected per class across the fusion folds. This helps avoid scenarios where classes get zero samples.

Table 10.1
UCM benchmark dataset

| Label | Class | Label | Class |
|---|---|---|---|
| **1** | agricultural | **12** | intersection |
| **2** | airplane | **13** | medium residential |
| **3** | baseball diamond | **14** | mobile home park |
| **4** | beach | **15** | overpass |
| **5** | buildings | **16** | parking lot |
| **6** | chaparral | **17** | river |
| **7** | dense residential | **18** | runway |
| **8** | forest | **19** | sparse residential |
| **9** | freeway | **20** | storage tanks |
| **10** | golf course | **21** | tennis court |
| **11** | harbor | | |

The reader should refer to our prior research for a detailed discussion of how to learn a ChI per class/output neuron versus a shared ChI across all neurons [5]. In this article, the goal is not to simply analyze the accuracy of ChI-based DCNN fusion, that was already demonstrated in [5]. Instead, our goal is to open the hood on the learned solutions and see if the XAI-ChI tools provide additional insight. Most figures reported in this section are illustrated using the "redblue" color map coding for visual simplicity–which means zero is blue, white is 0.5, and 1.0 is red. Furthermore, in each figure the NNs have been assigned the indices CaffeeNet=1, GoogleNet=2, ResNet50=3 and ResNet101=4. Table 10.2 reports the classification accuracy of the individual NNs and their fused result. As a further point of reference, we also ran the PatternNet data set and obtained the classification results

reported in Table 10.2 [X]. As Table 10.2 is clearly the more difficult case, with respect to fusion, we use the XAI-ChI tools on it for analysis.

Table 10.2
Accuracy of Individual NNs and their ChI Fusion

| | **CaffeeNet** | **GoogleNet** | **RN50** | **RN101** | **ChI** |
|---|---|---|---|---|---|
| UCM | 0.974 | 0.980 | 0.986 | 0.985 | 0.990 |
| PatternNet | 0.932 | 0.953 | 0.955 | 0.955 | 0.997 |

### 10.6.1 Measure-Centric Indices

Figure 10.2 is the Shapley index. We cannot report all results as there are five cross validation folds with respect to the NNs, three fusion folds per NN fold, and 21 object classes. However, the trends are more-or-less consistent across all folds in the UCM data set. As such, we arbitrarily selected the first fold.

Figure 10.2 reveals a few interesting stories. First, classes 13 and 19 clearly indicate the existence of a (different) dominant NN. On the other hand, classes 1, 2, 4, and 6 say all four NNs have equal worth. Overall, there are no global trends–i.e., no consensus across classes about a single best NN or a single poor NN. If we listen to the Shapley, it seems to say that different classes require different fusions and all NNs are vital to achieving success. Next, we analyze the interaction index (Figure 10.3).

In Figure 10.3, class 1 indicates independence between all of the NNs. However, class 15 indicates positive interaction between ResNet50 and ResNet101, and more-or-less independence otherwise. Class 10 indicates strong negative interactions (redundancies) be-

tween CaffeeNet and the other three networks. While it is possible to study each class across all folds, what is apparent is variation across the interaction index results. There does not appear to be a NN that we can remove–or a NN that we can solely rely on.

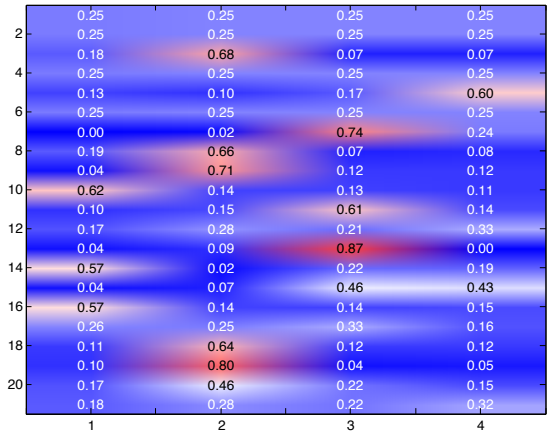



Figure 10.2
Visualization of the Shapley index values for (NN fold 1, fusion fold 1). Rows are the 21 classes and the x-axis is the four NNs. Each row sums to one.

### 10.6.2 Integral-Centric Indices

In this sub-section we investigate how the NNs are being combined via the ChI. Figure 10.4 is a visual summarization for NN fold 1 and fusion fold 1. Again, we focus on a single NN-fusion fold for sake of space.

The takeaway from Figure 10.4 is as follows. First, once again we see diversity in the way that different classes are aggregating their data. Classes 1, 2, 4, and 6 are for sure LCOS operators–as their fourth column distance values are zeros. However, they are not exactly min, max or mean like–confirmed by the other column one to three values. If anything, they are the most like a mean operator. What is interesting is there are numerous

non-LCOS like ChIs learned. This is encouraging per se because it helps to maybe justify that the ChI has a place here, versus a simpler fixed operator.

### 10.6.3 Data-Centric Indices

The summary of the last two sections is that there is variety in the worth of the individual NNs, their interactions and the aggregation operators. In this sub-section we explore the problem from a data-driven perspective to see if there is anything else to learn. Figure 10.5 shows variable visitation frequencies and percentage of data supported variables.

Figure 10.5 reveals a more disturbing trend in the UCM data set, as it relates to fusion. There is a strong imbalance in the visitation of variables and many low percentages of data supported variables. This indicates that variety is low in this data set–again, with respect to fusion–and as such we do not really have adequate data to support learning a quality fusion solution. Figure 10.6 reinforces this claim in terms of observed percentage of LCSs and dominant walk assessment.

Figure 10.6 is alarming in the respect that most learned ChIs see approximately only ten to forty percent of the different underlying ChI LCSs. Furthermore, the index for dominant walk indicates that most NNs have a single dominant walk, more support in the claim of low data diversity.

Based on the alarming evidence from the data-driven indices, we went back to the data and looked at both the outputs (decisions) from the different NNs and the miss-classification cases. What we discovered was two fold. First, the NNs are extremely strong classifiers, as indicated by their individual accuracy rates. As such, in the vast majority of

cases they are all "voting" the same way. Second, most of the missed chips are either naturally ambiguous chips or mislabeled chips–e.g., a golf course with no golf features that was called forest (which would likely trick a human). As such, when they get something wrong they all more-or-less get it uniformly wrong. There is not much to work with in terms of fusion. Overall, there are numerous red flags that indicate a scenario of poor training data to support learning quality fusion solutions.

On a side note, the dominant walk imbalance seems to be related to the fact that we are fusing very strong learners. Meaning, a great number of inputs result in the same value across the NNs, a confidence of $100\%$. As such, the default sort order becomes an ascending index ordering. Meaning, the dominant walk is really a product of the (too) strong agreement between the NNs and a further testament to low diversity.

### 10.6.4 Measure + Integral + Data Index Combination

In order to reach a holistic decision about the quality of our fusion solution, we need to combine the results of these three indices. The reality is these indices are obviously not independent of one another. The indices merely give us different ways to answer various questions. As such, the data-centric indices tell us that there is a severe lack of data diversity in the UCM data set. The Shapley, interaction index and aggregation indices tell us that there is diversity–meaning no obvious worthless DCNNs that can be removed. However, we would trust those indices a great deal if there was ideal data diversity. Since there is not, we must conclude that while we observe performance improvement on the

UCM data set, there is not really the data to support the learning of fusion and based on what we can observe all DCNNs are needed.

## 10.7 Conclusion and Future Work

In this article, we summarized existing indices and introduced new data-driven indices to support *explainable AI* (XAI) for the Sugeno Choquet integral. These indices were applied to the fusion of a set of heterogeneous architecture deep convolutional neural networks in remote sensing. The indices were used to assemble a more complete understanding of how fusion was working (or not working!).

The data-driven indices put forth, and how we ultimately "use them", are preliminary. For example, in our remote sensing problem there were many folds and classes. There are likely better indices and/or better methods of summarizing the findings across folds to build a more complete and informative picture. Also, we are currently using these indices in a manual investigative mode. Our desire is to find new algorithmic ways to use these indices to improve the results. For example, it might be possible to use these indices during training to promote less over-fitting and more diverse NNs.

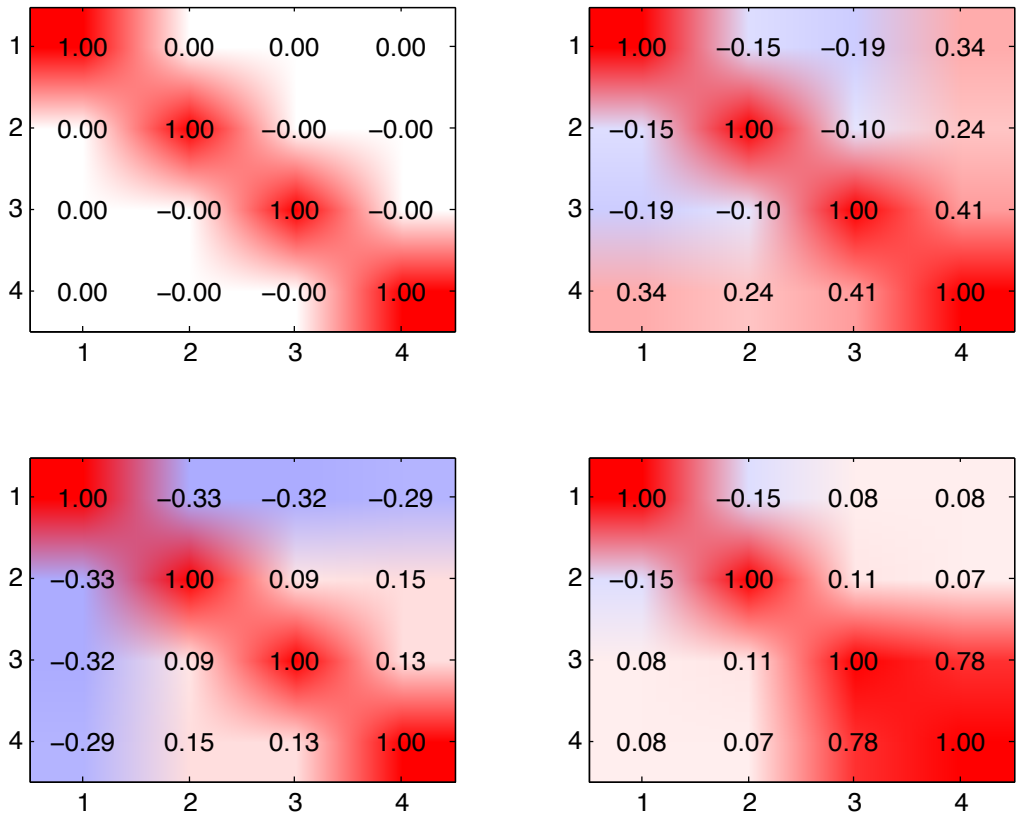



Figure 10.3
Visualization of the interaction index for (NN fold 1, fusion fold 1) and classes 1 (upper left), 5 (upper right), 10 (lower left) and 15 (lower right). Values range from -1 to 1.

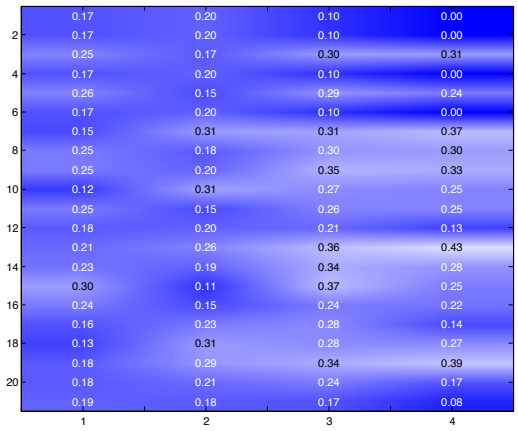



Figure 10.4
Visualization of the indices for introspection for (NN fold 1, fusion fold 1). Column one is $D_1$ (max), two is $D_2$ (min), three is $D_3$ (mean) and four is $D_4$ (LCOS).

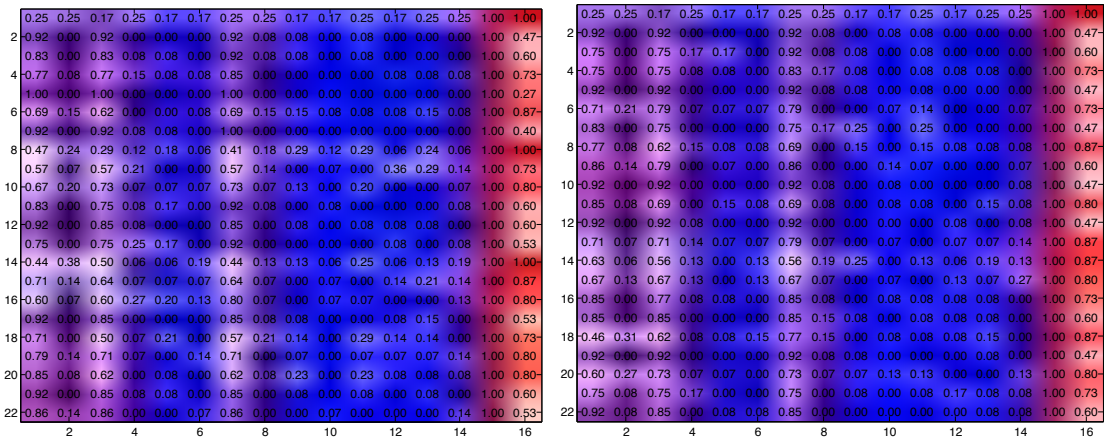
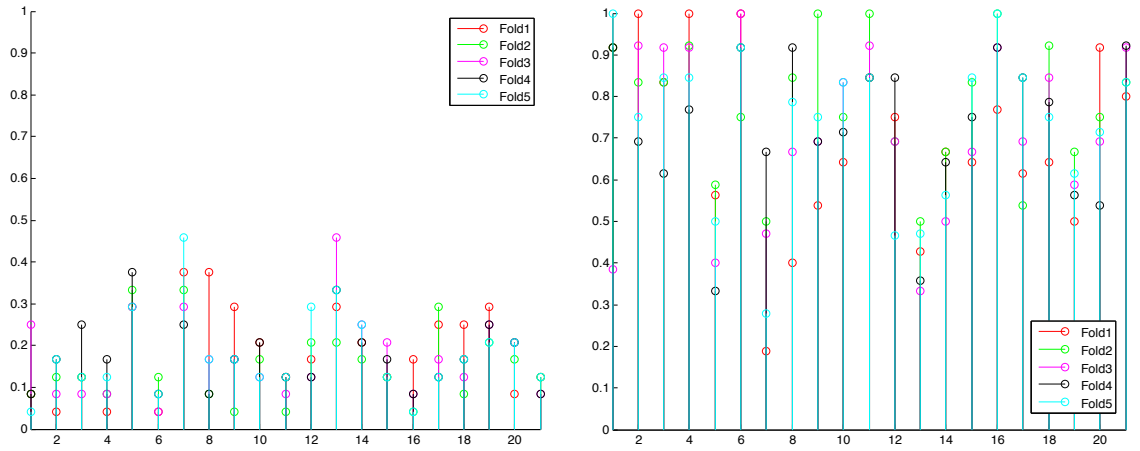


Figure 10.5

Visualization of variable visitation frequency and percentage of data supported variables for (left) (NN fold 1, fusion fold 1) and (right) (NN fold 2, fusion fold 1). Variables appear according their binary encoded index. Row 1 are ideal values–meaning, if the variable visitations were uniformly distributed then that is the value (and thus color) they should be. Row 2 is NN class 1, row 3 is NN class 2, and so forth. Columns 1 to 15 are the FM variable according their binary encoded index. For example, column 1 is variable $\{x_1\}$, column 2 is $\{x_2\}$, column 3 is $\{x_1, x_2\}$, and so forth. Column 16 is the percentage of data supported variables for each class (relative to color and decimal point display resolution).




(a) Percentage of LCS operators observed (value one desired)

(b) Dominant walk index (lower values preferred)

Figure 10.6

(a) Percentage of LCS operators observed and dominant walk index. X-axis is output neuron (class) number.

# CHAPTER XI

# FUZZY CHOQUET INTEGRATION OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR REMOTE SENSING

## 11.1 Introduction

We humans excel at many robust pattern recognition tasks in which computational systems can only perform well when limited in scope and constrained in operating environment. The human visual system is no exception. Humans develop at an early age a comprehensive visual processing and pattern recognition ability. Our vision allows us to process our physical environment (navigation) and facilitates many higher-level cognitive functions such as object classification and entity resolution. We accomplish this via a complex multistage visual system that begins with basic lightness and color receptors, then builds upon the perceived edges to derive shapes, spatial relationships, and eventually to organization of components into objects of interest – and this is before any higher level cognitive processing.

Deep neural network models follow a similar paradigm conceptually, extracting first edges and other simple geometric primitives in the lowest levels, then later mid-level assemblies of these primitives into visual concepts, which are then combined in higher-level layers as object components (blobs), that are eventually agglomerated into objects. These visual objects are agglomerated within fully connected neural layers for eventual classifi-

cations, which is an informational (cognitive) output. What deep architectures lack at the moment is the heterogeneous and dynamic capabilities of the human system, which is in part because a single architecture is not capable of the level of modeling and representation of the complex human system. Therefore, a heterogeneous set of pathways from sensory stimulus to cognitive function needs to be developed in a richer computational model. The model proposed in this chapter represents the learning of multiple pathways–as deep neural networks–coupled with appropriate information fusion. We feel fusion of the cognitive outputs (information) from multiple heterogeneous models (pathways) is the next step towards robust computational cognitive processing of visual, and visual-like, sensory data.

In general, *computational intelligence* (CI) is a branch of mathematics inspired by nature. Specifically, CI is associated with *neural networks* (NNs), *evolutionary algorithms* (EA) and *fuzzy set theory* (FST). NNs were established in 1943 by McCulloch and Pitts [134], FST was established in 1965 by Zadeh [208] and EAs were made popular by Holland in the early 1970s [94] (but arguably have roots going back as far as Turing in 1950). The point is, CI has existed in one form or another since the advent of *artificial intelligence* (AI). In this chapter, we focus on the intersection of NNs and FST for pattern recognition. In the last decade, substantial interest and effort has gone into *deep learning* (DL), a re-branding of NNs. This shift has forced us to re-address fundamental questions like; should humans design features (the classical approach to pattern recognition) or is a machine better at this task? Empirically, DL has more-or-less unanimously topped the charts in performance in many domains (e.g., natural language processing [48, 170], vision [60, 126, 45, 180, 46], remote sensing [29, 99, 41, 207]). However, while DL has generated

great excitement, much remains to be explored and explained. In this chapter, we focus on the specific question of how to perform decision-level fusion of DL networks.

DL can be viewed as a generalization of the classical pattern recognition pipeline– e.g., pre-processing, feature extraction (selection and/or reduction), classification and post-processing. In some settings this is now being called *shallow learning* because there are only a few "layers" in the pattern recognition pipeline. In the context of computer vision, DL can also be decomposed into levels; "low" (e.g., signal/image analysis via convolution), "mid" and "high" (more AI than signal processing, e.g., MLP classification). In the extreme, DL is nothing more than a series of operations that transform data to decisions. The point is, fusion can (and often does) take place at different levels in pattern recognition/DL. For example, keeping with the fusion nomenclature of the *Joint Directors of Laboratories* (JDL) [174], some fusion algorithms do *signal-in-signal-out* (SISO), whereas others do *feature-in-feature-out* (FIFO) and *decision-in-decision-out* (DIDO). If we regard DL as a SIDO process (e.g., SI=image and DO=class label), then it can be decomposed into its corresponding SISO, SIFO, FIFO, FIDO, DIDO (and combinations therein). In summary, fusion is not as simple as "cram data into a DL and let it do its thing".

Herein, we restrict our analysis to *deep convolutional neural networks* (DCNNs) [60, 126, 45, 180, 46, 209, 179], versus *auto encoders* (AEs) [92, 191, 40, 59, 64], *deep belief nets* (DBNs) [91, 128], etc., for sake of discussion tractability. The reality is, we still know little-to-nothing about fundamental DL fusion questions such as; (i) how/where is fusion currently happening, (ii) based on our current set of neurons/transformations, what is mathematically expressible and what is not (but should be), (iii) how should we be per-

forming fusion at different levels, (iv) how do we address heterogeneity with respect to semantics and/or uncertainty across data/information sources, and (v) how do we explain what fusion is doing (aka *explainable AI* (XAI)), to list a few. Independent of DL, fusion is a complicated topic that often means different things to different people in different fields (and even within the same field). Fusion is a wealth of challenges wrapped up into one term. Fusion ranges from data association (e.g., finding a one-to-one mapping between pixels in one sensor to pixels in another) to the mathematics of aggregation (specific functions/operators). In general, the idea of fusion is to obtain a "better" result than if we only used the individual inputs. However, better is not a well defined concept. In some applications, better might mean taking a set of inputs and reducing them into a single result that can be more efficiently or effectively used for visualization. Better could also refer to obtaining more desirable properties such as higher information content or lower conflict. In areas like pattern recognition, better often refers to some desirable property like more robust and generalizable solutions (e.g., classifiers). Regardless of the task at hand or the particular application, fusion is a core tool at the heart of numerous modern scientific thrusts.

In this chapter, we make the following contributions. First, we discuss two approaches for heterogeneous DCNN architecture fusion; density-based imputation and full *Choquet integral* (ChI) learning (per neuron and "shared weight"). Second, we outline indices for introspection and information theoretic indices to understand the capacity and integral (moving us closer to a so-called XAI solution versus black box solution). Third, we demonstrate and analyze these ideas on remotely sensed data. Fourth, we provide

open source code at www.derektanderson.com/FuzzyLibrary and https://github.com/scottgs/fi_library.

## 11.2 Deep Convolutional Neural Networks

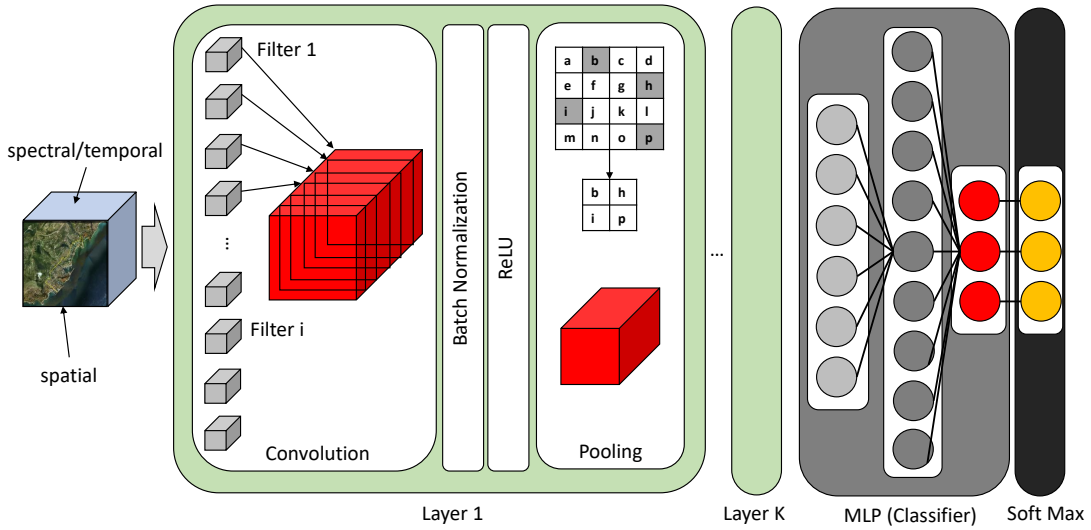



Figure 11.1
Example CNN. Input is a 3D cube (x-y are spatial, z is spectral), green layers consist of some subset of convolution (morphology, etc.), pooling (average, max, etc.), batch normalization (or other method to help mitigate overfitting) and nonlinear function (e.g., ReLU activation). The output of the green layers are typically fed to a MLP and optional post-processing steps (e.g., soft max normalization).

To date, the AE [92, 191, 40], CNN [60, 126, 45, 180, 46, 209, 179], DBN [91, 128] and *recurrent NNs* (RNNs) [139, 61] are the most mainstream DLs. However, other DL approaches exist, e.g., deep inference nets ([157] Verma et al. Takagi-Sugeno-Kang deep net), deconvolution CNNs (specifically transpose matrix convolution) [199, 210, 211] and morphological shared weight neural networks [198, 114]. Herein, we focus on the CNN, which is by far the most employed and often the highest performer. With re-

spect to the CNN, a number of architectures have been explored to date, e.g., AlexNet [126], GoogLeNet [179], VGGNet [166] and their derivatives. These architectures can be downloaded and extended (training, evaluation, visualization) via open source libraries like TensorFlow [1], CaffeNet [112], and MatConvNet [190]. The fundamental challenges of which architecture, how deep versus wide, hyperparameter tuning, what neuron types, how to transfer a DL from one domain to another (transfer learning [206]), and other questions are unanswered. Also, numerous associated challenges exist; e.g., lack of training data volume (and variety), class imbalance, dimensionality (spatial, temporal and spectral), explainable DL (what did the DL learn, versus a black box solution), to name a few. While DL has sparked a revolution in computer vision, pattern recognition and AI in general, an overwhelming number of theoretical and applied questions remain ripe for exploration.

In general, most CNNs consist of combinations of the following operations (see Figure 11.1). First, let the input to the system, $O_0$, be a three dimensional data cube of size $N_0 \times M_0 \times D_0$; where $N_0$ and $M_0$ are spatial dimensions and $D_0$ is the temporal or spectral dimensionality (e.g., RGB imagery has $D_0 = 3$). **(Convolution)** The backbone of a CNN is filtering via convolution. Filtering can take a number of meanings, e.g., enhancement, denoising or detection. Convolution specifics include factors like (i) stride (spatial and/or spectral/temporal "jumps") and (ii) padding (if no padding is used then the spatial dimension shrinks). **(Pooling)** Pooling is often applied to reduce spatial dimensionality–and combat challenges related to affine variation, noise, etc. Most often, average and max pooling are used. **(Activation)** Nonlinearity is also typically applied, in the form of a function like hyperbolic tangent (*tanh*), sigmoid, or ReLU ($ReLU(x) = \max(0, x)$). **(Training**

**Techniques)** In order to combat factors like sensitivity to parameter selection and over-training, methods like dropout [173], regularization [68] and/or batch normalization [103] (addresses internal covariate shift and vanishing gradients) are often used. Beyond architecture, there are factors like GPU acceleration [35], training (e.g., *stochastic gradient decent* (SGD) [33], SGD with momentum [153, 178], AdaGrad [57], RMSProp [186] and ADAM [123]). The reader can refer to [68] for additional mathematical and algorithm details related to CNNs. The reader can also refer to [21] for a recent survey of DL in remote sensing (theory, applications and open challenges).

The idea of FST in NNs is not new. The reader can refer to the work of Pal and Mitra [148] for neuro-fuzzy pattern recognition. Pal, Mitra, and others (e.g., Keller and the fuzzy perceptron [119]), explored a variety of topics such as fuzzy min-max networks, fuzzy MLPs, and fuzzy Kohonen networks. In terms of aggregation, a few FST works have been explored to date. In 1992 [202], Yager put forth the *ordered weighted average* (OWA) [201]–which technically is a *linear combination of order statistics* (LCOS) since the weights are real-valued numbers (versus sets)–neuron. In 1995, Sung-Bae utilized the OWA for NN aggregation (at the decision/output level) [177]. In 1995, Sung-Bae et al. also explored the fuzzy integral, the Sugeno fuzzy integral not Sugeno's fuzzy ChI, for NN aggregation [43]. Specifically, they used the Sugeno $\lambda$-*fuzzy measure* (FM) and the densities were derived using their respective accuracy rates on training data. In 2017 [164], we (Scott et al.) used the Sugeno and ChIs for DCNN fusion. Specifically, Scott et al. used transfer learning to adapt GoogLeNet, AlexNet and ResNet50 from perspective RGB imagery to aerial remote sensing imagery. Scott then applied different aggregations–

the fuzzy integral, voting, arrogance, and weighted sum–to these DCNNs. Scott's fusion was based on the Sugeno $\lambda$ FM and the densities were (i) set to the DL normalized classifier accuracies and (ii) a genetic algorithm was used to learn the densities (which led to higher performance).

## 11.3 Fuzzy Measure and Fuzzy Integrals

The ChI has been successfully demonstrated in numerous applications; e.g., explosive hazard detection [155, 167, 168], computer vision [181], pattern recognition [77, 80, 137, 118, 62], remote sensing [162], multi-criteria decision making [70, 127], forensic anthropology [6, 8, 14], control [187], multiple kernel learning [152, 150, 155, 98, 97], multiple instance learning [55], ontologies [3], missing data [110], and most relevant to the current chapter, DL [164]. The ChI is a nonlinear aggregation function that is parameterized by the FM (aka capacity). Countless mathematical variations of the fuzzy integral have been put forward for different reasons; e.g., address different types (i.e., real-valued, interval-valued, set-valued) of uncertainty in the integrand and/or measure, limit the number of input interactions for tractability, etc. Herein, we focus on and succinctly review just the real-valued discrete (finite $X$) ChI for DCNN fusion.

### 11.3.1 Discrete (Finite $X$) Fuzzy Measure

Let $X = \{x_1, x_2, ..., x_N\}$ be $N$ sources, e.g., experts, sensors, or in the case of this chapter, DCNNs. The first action we face is how to assign "worth/utility" to different subsets of DCNNs. For example, the well-known backbone of calculus on real-valued domains is the Lebesgue measure; which coincides with length, area and hypervolume.

However, when $X$ is a discrete domain, e.g., set of DCNNs, what is the corresponding "measure"? In [181], Keller et al. first investigated the idea of using the fuzzy integral for pattern recognition. A FM is a function, $\mu$, on the power set of $X$, $2^X$, which satisfies (1) (boundary condition) $\mu(\emptyset) = 0$ and (2) (monotonicity) if $A, B \subseteq X$ and $A \subseteq B$, then $\mu(A) \leq \mu(B)$. Often, the constraint $\mu(X) = 1$ is imposed in settings like decision level fusion[1].

### 11.3.2 Discrete (Finite $X$) Fuzzy Integral

The FM models important "interactions" (e.g., subjective worth, statistical correlation, etc.) between different source subsets. The input provided by our sources is $\{h(\{x_1\})$ $, h(\{x_2\}), ..., h(\{x_N\})\}$. The fuzzy integral is a way to combine the integrand ($h$) information relative to the FM ($\mu$). Let $h(\{x_i\}) \in \Re^{\geq 0}$ be the data/information from source $i$. The discrete (finite $X$) Sugeno FI is[2]

$$\int_S h \circ \mu = S_\mu(h) = \bigvee_{i=1}^{N} \big(h(\{x_{\pi(i)}\}) \wedge \mu(A_i)\big), \tag{11.1}$$

where $\pi$ is the permutation $h(\{x_{\pi(1)}\}) \geq h(\{x_{\pi(2)}\}), \ldots, \geq h(\{x_{\pi(N)}\})$, $A_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$ and $\mu(A_0) = 0$. The discrete (finite $X$) ChI is[3]

[1] If $\mu(X) < 1$, properties like idempotency and boundedness are not guaranteed.

[2] Due to the maximum (t-conorm) and minimum operators (t-norm), the Sugeno FI does not actually generate any possible number between the minimum and maximum of the inputs. Instead, it selects one of the FM or input values, i.e., at most one of $2^N + N$ values.

[3] The ChI is used frequently for a number of reasons; e.g., it is differentiable [137], for an additive (probability) measure it recovers the Lebesgue integral, it yields a wider spectrum of values between the minimum and maximum inputs (versus the discrete and relatively small number of values that the Sugeno FI selects from), etc.

$$\int_C h \circ \mu = C_\mu(h) = \sum_{i=1}^{N} h(\{x_{\pi(i)}\}) \left[\mu(A_i) - \mu(A_{i-1})\right]. \tag{11.2}$$

Since the ChI is a parametric aggregation function, once the FM is determined the ChI turns into a specific operator. For example: if $\mu(A) = 1, \forall A \in 2^X \setminus \emptyset$, the ChI becomes the maximum operator; if $\mu(A) = 0, \forall A \in 2^X \setminus X$, we recover the minimum; and if $\mu(A) = \frac{|A|}{N}$, we recover the mean. Each of these cases can be viewed as constraints or simplifications on the FM (and therefore the ChI). In general, the discrete ChI has $N!$ unique input sortings and each yields a linear convex sum operator.

### 11.3.3 Data-Driven Optimization

The first challenge we must confront is, where do we get the FM ($\mu$) from? One option is to have an expert specify it. However, this is not practical (assuming the expert could even meaningfully assign values to the interactions) as the number of inputs (e.g., DLs) increases. Another option is we can specify or try to learn the worth of just the singletons (the densities). From there, a number of formulas can be used to impute (fill in) the missing variable values. Popular approaches include the Sugeno $\lambda$-FM and the S-Decomposable FM [176]. However, while convenient, most often we do not obtain the desired values for variables that we need. With respect to pattern recognition, the focus of this chapter, another route is to learn it from data. Next, we review one way to learn the FM, and therefore the ChI, in the context of DIDO for DL. However, the reader can refer to [111] for an efficient learning method with only data-supported variables and [121] for a review of alternative FM/ChI learning methods.

We quickly summarize one way to learn the full FM/ChI (see [12] for full mathematical explanation). Let $O = \{\mathbf{h}_j, y_j\},\ j = 1, 2, \ldots, M$, be $M$ training examples; where $\mathbf{h}_j$ is the $j$-th instance with data from $N$ inputs and $y_j$ is the ground-truth for $\mathbf{h}_j$. The sum of squared error for training dataset $O$ is

$$E(O, \mathbf{u}) = \sum_{j=1}^{M} e^j = \sum_{j=1}^{M} (\mathbf{c}_j^T \mathbf{u} - y_j)^2 = ||D\mathbf{u} - \mathbf{y}||_2^2, \tag{11.3}$$

where $\mathbf{u} = [\mu(\{x_1\}), ..., \mu(\{x_N\}), \mu(\{x_1, x_2\}), \mu(\{x_1, x_3\}), ..., \mu(X)]$ (lexiographic vector of size $2^N - 1$), $\mathbf{c}_j$ holds the coefficients of $\mathbf{u}$ for observation $\mathbf{h}_j$ (e.g., for $N = 3$ and $h(\{x_2\}) \geq h(\{x_1\}) \geq h(\{x_3\})$, $c = [0, h(\{x_2\}) - 0, 0, h(\{x_1, x_2\}) - h(\{x_2\}), 0, 0, 1 - h(\{x_1, x_2\})]$), $D = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_M]^T$ (full dataset), $\mathbf{y} = [y_1\ y_2\ \ldots\ y_M]^T$, and $||\cdot||_2$ is norm-2 operation, $\mathbf{u}$. The regularized SSE optimization problem is

$$\min_{\mathbf{u}} f(\mathbf{u}) = ||D\mathbf{u} - \mathbf{y}||^2 + \beta v(\mathbf{u}), \tag{11.4}$$

where $\beta \in \Re^{\geq 0}$ (regularization constant, which balances the "cost" (or penalty) of obtaining minimum function error relative to our desire to have minimal model complexity) and $v(\mathbf{u})$ is an index of model complexity (e.g., k-additive and Mobius, Gini-Simpson, $\ell_p$-norm, etc. [149]), subject to the FM boundary and monotonicity conditions (see [12] for how to pack the constraints into a linear algebra expression), which can be solved via quadratic programming. Full code and explanation (including how to build the constraint matrix $C$) can be found at `www.derektanderson.com/FuzzyLibrary` and `https://github.com/scottgs/fi\_library`.

### 11.3.4 Explainable AI (XAI) Fusion

It is one thing to train a network and another to understand it! In this subsection, we highlight FM and ChI indices for the purpose of *explainable AI* (XAI)[4]. XAI is an attempt to explain the inner operations of pattern recognition for purposes like describing it to others for domain knowledge transfer, trust, etc. The Shapley index addresses the importance or worth of each input (aka DL),

$$\Phi_\mu(i) = \sum_{K \subseteq X \backslash \{i\}} \zeta_{X,1}(K) \left( \mu(K \cup \{i\}) - \mu(K) \right), \tag{11.5}$$

where $\zeta_{X,1}(K) = \frac{(|X|-|K|-1)!|K|!}{|X|!}$, $K \subseteq X \backslash \{i\}$ denotes all proper subsets from $X$ that do not include source $i$. The Shapley value of $\mu$ is the vector $\Phi_\mu = (\Phi_\mu(1), ..., \Phi_\mu(N))^t$ and $\sum_{i=1}^{N} \Phi_\mu(i) = 1$. The Shapley index can be interpreted as the average amount of *contribution* of source $i$ across all coalitions. The next index informs us about how two inputs interact with one another (aka what advantage is there in combining DLs). The interaction index (Murofushi and Soneda [142]) between $i$ and $j$ is

$$\mathcal{I}_\mu(i,j) = \sum_{K \subseteq X \backslash \{i,j\}} \zeta_{X,2}(K)(\mu(K \cup \{i,j\}) - \mu(K \cup \{i\}) - \mu(K \cup \{j\}) + \mu(K)), \quad i = 1, ..., N, \tag{11.6}$$

where $\zeta_{X,2}(K) = \frac{(|X|-|K|-2)!|K|!}{(|X|-1)!}$, $\mathcal{I}_\mu(i,j) \in [-1,1], \forall i,j \in \{1,2,...,N\}$. A value of $1$ (respectively, $-1$) represents the maximum complementary (respective redundancy) between $i$ and $j$. The reader can refer to [78] for further details about the interaction index, its con-

[4]Implementation can be found at `www.derektanderson.com/FuzzyLibrary` and `https://github.com/scottgs/fi\_library`

nections to game theory and interpretations. Grabisch later extended the interaction index to the general case of any coalition [74],

$$\mathcal{I}_\mu(A) = \sum_{K \subseteq X \setminus A} \zeta_{X,3}(K, A) \sum_{C \subseteq A} (-1)^{|A \setminus C|} \mu(C \cup K), \quad i = 1, ..., N, \tag{11.7}$$

where $\zeta_{X,3}(K, A) = \frac{(|X|-|K|-|A|)!|K|!}{(|X|-|A|+1)!}$. Equation (11.7) is a generalization of both the Shapley index and Murofushi and Soneda's interaction index as $\Phi_\mu(i)$ corresponds with $\mathcal{I}_\mu(\{i\})$ and $\mathcal{I}_\mu(i, j)$ with $\mathcal{I}_\mu(\{i, j\})$.

The above indices are focused strictly on the FM. A different fundamental type of question is what "type" of aggregation is the ChI performing? Answering this question helps us understand how the DL information is being combined (e.g., in an optimistic, pessimistic, expected value like fashion, etc.). In [154], we established an index ($D_1$) to measure the degree to which a given FM/ChI is an maximum operator. In the following, we discuss the FM in terms of its underlying lattice structure. Let "layer k" (measure defined on sets of cardinality $k$) be denoted by $L(k)$, e.g., $L(1) = \{\mu(\{x_1\}), \mu(\{x_2\}), \mu(\{x_3\})\}$ for $N = 3$. Next, let $\mathbf{W} = \frac{[\frac{1}{N}, ..., 1]}{\sum_{i=1}^{N} \frac{i}{N}}$ be weights (penalty or costs) for each layer and

$$D_1(\mu) = \sum_{k=1}^{1} \frac{\mathbf{W}(k)}{2}(T_1 + T_4) + \left[ \sum_{k=2}^{N} \frac{\mathbf{W}(k)}{3}(T_1 + T_2 + T_4) \right], \tag{11.8}$$

$T_1 = 1 - \left( \frac{\sum_{I \in L(k)} \mu(I)}{|L(k)|} \right)$, $T_2 = \left( \frac{\sum_{I \in L(i)} \mu(I)}{|L(k)|} - \frac{\sum_{J \in L(k-1)} \mu(J)}{|L(k-1)|} \right)$, $T_3 = \frac{\sum_{I \in L(k)} \mu(I)}{|L(k)|}$ and $T_4 = \frac{\sum_{I \in L(k)} (\mu(I) - T_3)^2}{|L(k)|-1}$. A value of $D_1 = 0$ means the ChI is the maximum operator. The distance of a learned capacity to a minimum operator ($D_2$), mean ($D_3$) and LCOS ($D_4$), for $\mathbf{W}_2 = \frac{\left[1, ..., \frac{1}{N-1}\right]}{\sum_{i=1}^{N-1} \frac{i}{N-1}}$, is

$$D_2(\mu) = \sum_{k=1}^{1} \frac{\mathbf{W}_2(k)}{2} (T_3 + T_4) + \left[ \sum_{k=2}^{N-1} \frac{\mathbf{W}_2(i)}{3} (T_3 + T_2 + T_4) \right], \tag{11.9}$$

$$D_3(\mu) = \frac{1}{2^N - 2} \sum_{k=1}^{N-1} \sum_{I \in L(k)} \left| \mu(I) - \frac{k}{N} \right|, \tag{11.10}$$

$$D_4(\mu) = \frac{1}{N-1} \sum_{k=1}^{N-1} \sqrt{T_4}. \tag{11.11}$$

## 11.4 DCNN Fusion Based on Fuzzy Integration

The focus of this chapter is the fusion of different state-of-the-art DCNN architectures. However, the procedures outlined herein are applicable to other neural inputs (see Figure 11.2).

### 11.4.1 DCNN Architectures Used for Fusion

The first NN used herein for fusion is CaffeNet [112], which is a derivative of AlexNet with similar structure, except that the order of pooling and normalization is reversed to reduce learnable parameters. CaffeNet contains five convolutional feature extraction steps and three fully connected layers for classification. Classification is performed with two fully connected inner product layers and a final soft-max layer for the network output. The output of the soft-max classification layer is effectively a classification vector. CaffeNet represents the most simple and shallow of our DL investigated herein.

GoogLeNet [179] is a much deeper NN than CaffeNet–it has 27 parameterized layers. Because of this network depth, GoogLeNet has three classification outputs at various stages

of the network to facilitate error back propagation. GoogLeNet's novel *inception layer* processes the input with max-pooling, 1x1, 3x3, and 5x5 convolutions simultaneously in a feature extraction step, and the outputs are concatenated as the layer output to achieve a multi-scale feature extraction. Using multiple convolutions at each stage follows the intuition that features from different kernel scales can be extracted and processed at the same time, thereby extracting multi-scale visual features. GoogLeNet is from a family of networks commonly referred to as *inception networks*.

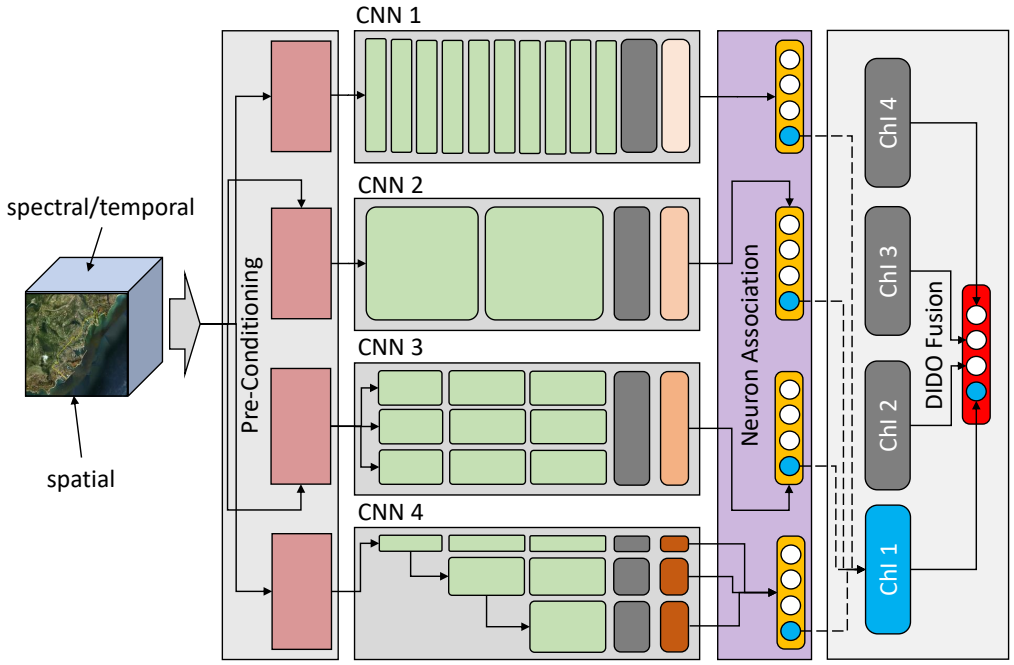



Figure 11.2
Illustration of DIDO DCNN fusion. Note, many possibilities exist; e.g., variations in architecture, pre-conditioning/transforms (e.g., conversion to frequency analysis versus spatial domain, band selection or grouping, etc.), training data, etc. Next, neuron mapping/association is required followed by aggregation. Herein, a different fusion operator is learned per output neuron (versus shared fusions/weights).

ResNet [90] is a collection of DCNN architectures inspired by VGGNet [166]. In both ResNet and VGGNet, the primary kernels used to construct the convolution layers are 3x3. The architecture design incorporates the following rules to govern their structure. First, if the output of the feature map is the same, then the same number of 3x3 convolutional layers will be used. Second, if the output of the feature map is halved, then it will use twice as many 3x3 convolutional kernels The ResNet architectures employ residual connections that bypass two or more convolution layers at a time, allowing error to better propagate backward through the network. These are commonly referred to as *residual networks*, and here the ResNet50 and ResNet101 architectures are used within our experimental design. These networks have 50 and 101 feature extraction steps, respectively.

### 11.4.2 Transfer Learning, Neuron Association and Conditioning

If we design a set of custom DCNNs then it is trivial to ensure a bijection (one-to-one and onto) output neuron mapping. However, if existing community pretrained DCNNs (GoogLeNet, AlexNet, etc.) are leveraged–a task encountered frequently in practice–then this is not guaranteed. One way to resolve the one-to-one mapping task is to replace and retrain the DCNN classification layers per the labels for the task at hand. This is a type of transfer learning that keeps the feature layers intact. In [163], we (i) replaced and retrained the classification layers and we also (ii) updated the feature weights (e.g., convolution layers). Thus, we built custom classifiers for remote sensing of aerial imagery based on a network initialized by ground-perspective RGB imagery. In addition, data augmentation via rotation and image flipping was applied as well. However, we remark that other avenues

exist; e.g., one could manually resolve the mapping or use an automated method based on an ontology. Regardless, using multiple custom or pretrained networks of different architectures raises another question; are the outputs numerically (e.g., all in $[0, 1]$) and semantically "to scale" (e.g., does $a$ (e.g., $a = 0.5$) in domain $i$ map to $a$ in the other domains). One way to mitigate this issue in practice is to add a soft max normalization (aka normalized exponent function) layer after the raw neuronal output layer. For example, if $\eta$ is the soft max output for neuron $o_j$ then the soft max function is $\eta(o_j) = \frac{e^{o_j}}{\sum_{n=1}^{N} e^{o_n}}$. Thereby, we bound the domain of input for the subsequent fusion layer of our pattern recognition system, ensure the data across networks and neurons is well conditioned.

### 11.4.3 Non-Optimization Approach: $\lambda$-FM Based Imputation of the ChI

The first fusion approach explored here is to exploit our knowledge about the performance of the individual DCNNs on training data [43, 164]. A classical approach to obtaining the remaining $2^N - 2 - N$ FM values (beyond the densities) is the Sugeno $\lambda$-measure. For sets $A, B \subseteq X$, such that $A \cap B = \phi$,

$$\mu_\lambda(A \cup B) = \mu_\lambda(A) + \mu_\lambda(B) + \lambda\mu_\lambda(A)\mu_\lambda(B), \tag{11.12}$$

for some $\lambda > -1$. In particular, Sugeno showed that $\lambda$ can be found by solving

$$\lambda + 1 = \prod_{i=1}^{N} (1 + \lambda\mu(x_i)), \lambda > -1, \tag{11.13}$$

where there exists exactly one real solution such that $\lambda > -1$. Some advantages of the Sugeno $\lambda$-measure include its simplicity, the $N$ densities can be more tractable to acquire,

fewer number of parameters can help address overfitting (versus using the full $2^N$ variables), and it is a probability measure when $\lambda = 0$. However, there is no guarantee in practice that the values that it imputes are what we actually need. Simply speaking, more information or a different imputation formula may be required; e.g., the S-Decomposable imputation formula, $\mu(A) = \bigvee_{i \in A}(\mu(x_i))$ (where $\bigvee$ is a t-conorm). Algorithm (6) describes how to use the Sugeno $\lambda$-measure to fuse a set of pretrained DLs based on individual performance for density.

**Algorithm 6:** $\lambda$-FM Based Imputation of the ChI from a Set of Pre-Trained DCNNs

**INPUT**: $\text{DL}_i$ - $N$ DCNNs ($B$ neurons each); $\bar{O}$ - labeled training data

1. Run each DCNN on $\bar{O}$, get *overall accuries* (OA); $a_{b,i} \in [0, 1]$ (i.e., performance of DL $i$ on class $b$)

2. Assign the $i$th density its corresponding OA; i.e., $\mu_{\lambda_b}(x_i) = a_{b,i}$

3. Find $\lambda_b$ (using $\{\mu_{\lambda_b}(\{x_1\}), ..., \mu_{\lambda_b}(\{x_N\})\}$) for the Sugeno $\lambda$-FM (aka solve Eq. (11.13))

4. Recursively calculate $\mu_{\lambda_b}(A), \forall A \in 2^X \setminus \{\{x_1\}, ..., \{x_N\}\}$, using the densities and $\lambda_b$ (Eq. (11.12))

**OUTPUT**: $B$ full fuzzy measures - $\{\mu_{\lambda_1}, ..., \mu_{\lambda_B}\}$

### 11.4.4 Optimization Approach: Learning the Full ChI

As stated in Section 11.4.3, there is no guarantee that imputation from densities results in the input interactions that we desire (and thus results in an appropriate aggregation operator). Algorithm (5) shows how to use quadratic programming for acquisition of the full FM for DIDO fusion of DCNNs (Algorithm (5) is how to learn a single "shared" FM to be applied to all neurons). Thus, training data is directly used to learn these crucial interactions–which means better selection of appropriate aggregation operator. However, as we discuss in [111], this process can lead to a big boost in performance but it is not without flaw. Specifically, in [111] we show that training data only typically supports a subset of FM variables. In return, we put forth an extended optimization of the ChI by (1) identifying which variables are supported by data, (2) optimizing just those variables and then (3) looking at imputation methods to infer the value of data unsupported variables based on application specific criteria. We do not have space to go into depth about the extension here, the reader can refer to [111] for full details.

## 11.5 Experiments

In this chapter, two benchmark remote sensing datasets suitable for classification tasks of objects or land-cover/land-use are used. Remote sensing data represents a significant pattern recognition challenge. As can be seen in Figures 11.3 and 11.9 below, the variability and complexity of overhead imagery is immense. The visual cues exist at multilple levels: fine-scale (e.g. airplane shapes, vehicle presence, etc.) to large-scale (e.g., road way configurations in overpasses versus intersections versus freeway). In fact the entire

**Algorithm 7:** Learn a Full FM/ChI Per Class for a Set of Pre-Trained DCNNs

1 **INPUT**: $\text{DL}_i$ - $N$ DCNNs ($B$ neurons each); $\bar{O}$ - labeled training data; $\beta$ - regularization value

2 1. Per class/output neuron ($b$), run each instance ($1 \leq j \leq |\bar{O}|$) through each DCNN ($i$); get $h_j^b(x_i)$ terms

3 2. Per neuron ($b$), construct the individual $D_b$ from the $h_j^b(x_i)$ terms

4 3. Run $B$ independent quadratic programs (on the $D_b$ respectively); yielding $\{\mu_1, ..., \mu_B\}$

5 **OUTPUT**: $B$ full fuzzy measures - $\{\mu_1, ..., \mu_B\}$

**Algorithm 8:** Learn a Single "Shared Weight" Full FM/ChI for a Set of Pre-Trained DCNNs

1 **INPUT**: $\text{DL}_i$ - $N$ DCNNs ($B$ neurons each); $\bar{O}$ - labeled training data; $\beta$ - regularization value

2 1. Per class/output neuron ($b$), run each instance ($1 \leq j \leq |\bar{O}|$) through each DCNN ($i$); get $h_j^b(x_i)$ terms

3 2. Per neuron ($b$), construct the individual $D_b$ from the $h_j^b(x_i)$ terms

4 3. Use quadratic program to solve $(||D_1\mathbf{u} - \mathbf{y_1}||_2^2 + ... + ||D_B\mathbf{u} - \mathbf{y_B}||_2^2 + \beta v(\mathbf{u}))$; yields $\mu$

5 **OUTPUT**: Full fuzzy measure - $\mu$

field of photo-interpretation revolves around developing human expertise in this pattern recognition task. For each of the datasets herein DCNNs were trained using the techniques detailed in [163], including transfer learning and data augmentation via rotation and image flipping. The trained DCNNs are then used in a locked state, i.e., no further learning happens in DL during the fusion stage. The training of the DCNNs are done in five-fold, cross validation manner; such that we have 5 sets of 80% training and 20% testing for both datasets. Per DCNN fold, three-fold CV fusion is used (due to limited data).

### 11.5.1 UC Merced (UCM) dataset

The UC Merced (UCM) benchmark dataset [145][204] has been used in a wide range of remote sensing research, including prior work in classification of objects and land-cover such as [164], [163], and [38]. Figure 11.3 shows exemplar image chips from each class of the UCM dataset. The dataset includes 21 classes that are a mix of objects (airplane, baseball diamond, etc.) and landcover (beach, chaparral, etc.). We see that some classes, e.g., harbor and parking lot, are complex compositions of sub-entities (boats and vehicles); while others are general structural patterns of shapes (e.g., intersection and baseball diamonds).

Table 11.1 is the result of fusion on the UCM dataset. First, we see that aggregation outperforms no aggregation (i.e., the individual DCNNs) in four out of five folds. Second, we see that min, max and average (basic aggregation operators) do well in comparison to the ChI. However, these three operators are specific instances of the ChI, which informs us that there are challenges with variety and thus generalizability of this particular data set

(otherwise they should have been selected). Next, it is interesting to see that the shared weight fusion solutions do as well as they do. It is our suspicion–something to be explored in future work–that a shared FM for the ChI helps combat overfitting. It is also our suspicion–again, subject of future work–that while the Sugeno $\lambda$-FM would not be our first choice, it might also help combat overfitting as it has just $N$ parameters versus the otherwise $2^N - 1$. However, the performance of the individual DCNNs (which were used as the densities) are so remarkably high that ultimately this forces the Sugeno $\lambda$-FM to more-or-less be the maximum operator.

Figure 11.3
Sample image chips from the 21 class UCM benchmark dataset, each 256x256 pixels approximately 0.3m *ground sampling distance* (GSD) spatial resolution. Classes in left-to-right, top-down order: 1 *agricultural*, 2 *airplane*, 3 *baseball diamond*, 4 *beach*, 5 *buildings*, 6 *chaparral*, 7 *dense residential*, 8 *forest*, 9 *freeway*, 10 *golf course*, 11 *harbor*, 12 *intersection*, 13 *medium residential*, 14 *mobile home park*, 15 *overpass*, 16 *parking lot*, 17 *river*, 18 *runway*, 19 *sparse residential*, 20 *storage tanks*, and 21 *tennis court*. In subsection 11.5.1, neuron indices are used instead of text descriptions for sake of compactness.

Table 11.1
Fusion Results for the UCM dataset

| **Accuracy** | **Method** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Full ChI Per Neuron | Full ChI Shared | SLFM ChI Shared | CaffeNet | GoogLeNet | ResNet 50 | ResNet 101 | Max | Avg | Min |
| **Fold 1** | 0.979 | 0.977 | **0.984** | 0.957 | 0.957 | **0.985** | 0.973 | 0.978 | 0.981 | 0.976 |
| **Fold 2** | 0.991 | **0.994** | **0.993** | 0.964 | 0.983 | 0.978 | 0.988 | **0.993** | **0.994** | **0.993** |
| **Fold 3** | 0.994 | 0.990 | 0.996 | 0.971 | 0.985 | 0.992 | 0.988 | 0.996 | 0.996 | **0.998** |
| **Fold 4** | 0.992 | 0.996 | 0.996 | 0.988 | 0.980 | 0.983 | 0.988 | 0.996 | 0.992 | **0.998** |
| **Fold 5** | **0.989** | 0.985 | **0.989** | 0.976 | 0.973 | 0.983 | 0.980 | **0.989** | **0.989** | 0.986 |

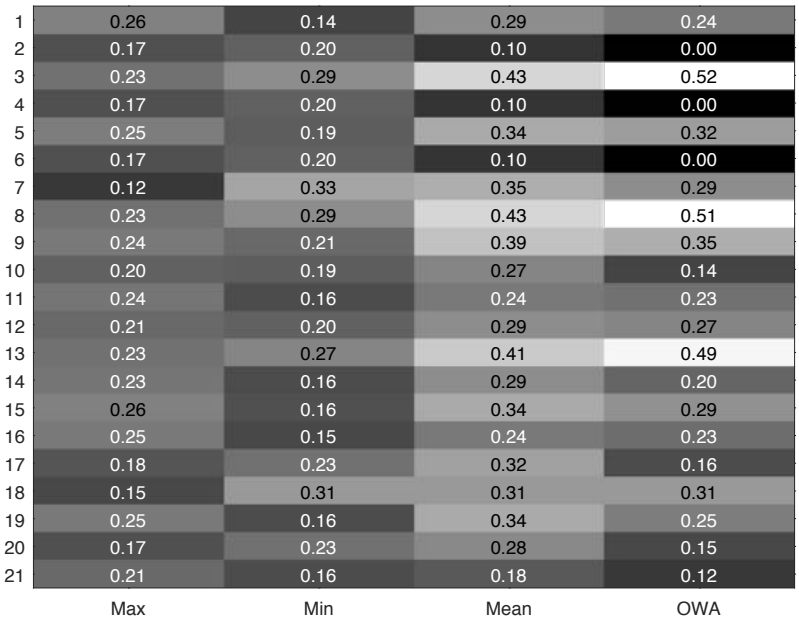
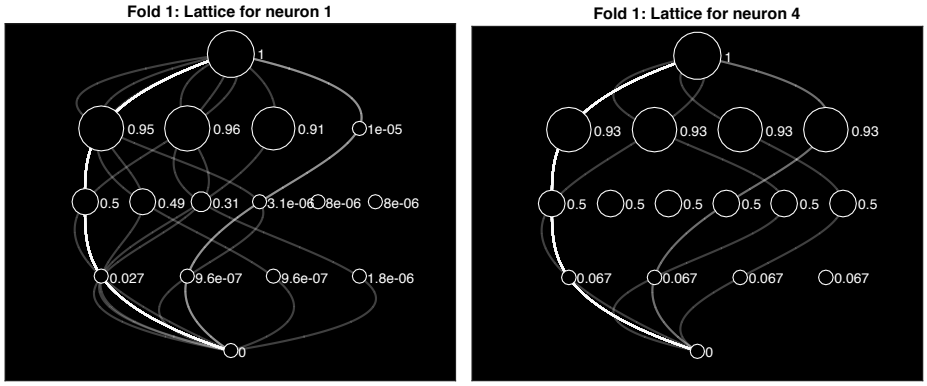



Figure 11.4
Color coded matrix showing the distances obtained using the four reported indices of introspection ($D_1(\mu)$ to $D_4(\mu)$) relative to the learned full ChI per neuron on fold 1 of the UCM dataset. y-axis is the neuron index (see Figure 11.3) and x-axis is the distance measure. Neurons two, four and six are OWA operators (but not min, max or mean like).

Next, Figure 11.4 gives us a feel for what type of aggregation strategy is being used for the 21 classes. Again, the max, min and mean are all OWAs, so we can start first with analyzing column four. There are three neurons (2, 4 and 6–i.e., airplane, beach and

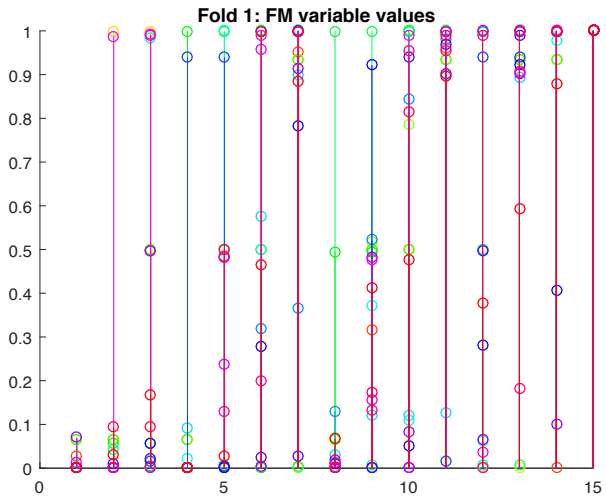
chaparral) that learned an OWA. The other neurons have learned something more unique, which helps justify the inclusion of the ChI versus say a simpler operator (see Figure 11.5(a)). At that, none of the learned OWAs are that similar to our extreme markers of max (a t-conorm or union like operator), min (a t-conorm or intersection like operator) or average (an expected value like operator). For example, Figure 11.5(b) shows one of these OWA operators, which breaks down into a trimmed mean operator.



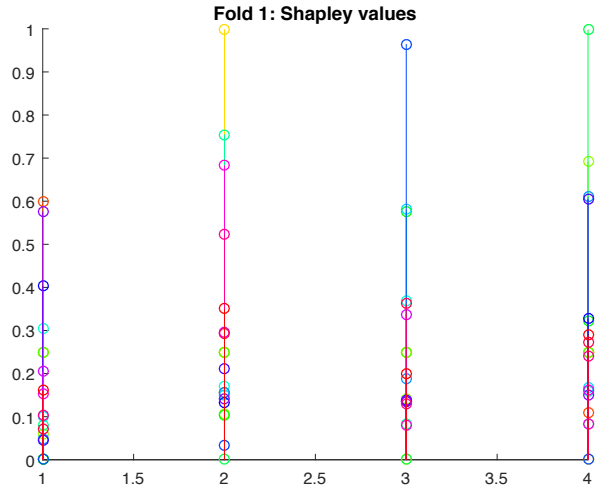

(a) (b)

Figure 11.5

Example of two full FMs for the (a) first and (b) fourth neuron in fold 1 of the UCM dataset. "Layer" $l$ (from bottom to top) in the image denotes FM variables with cardinally $l$. Thus, layer $0$ (bottom node) is the empty set, the next layer is the singletons, top is $\mu(X)$, etc. Each variable is presented in lexicographic order, i.e., layer $2$ is $\{x_1, x_2\}$, $\{x_1, x3\}$, $\{x_1, x_4\}$, $\{x_2, x_3\}$, $\{x_2, x_4\}$ and $\{x_3, x_4\}$. The nodes are also drawn size-wise proportional to their value (a minimum size and maximum was specified to make them still show up for $0$ valued variables). In addition, the "paths" drawn indicate the visitation frequency (the brighter the line, the higher the visitation) for the test data in fold 1. Furthermore, the fourth neuron learned an OWA with weights $(0.067, 0.433, 0.43, 0.07)^t$–a trimmed mean operator. Conversely, neuron one is more complex to decode. It does not reduce into a single compact description like an OWA. However, we can view it in terms of the $N!$ walks (possible sorts). Since the $h(\{x_1\}) \geq h(\{x_2\}) \geq h(\{x_3\}) \geq h(\{x_4\})$ is encountered frequently, we decode and analyze its weights. The linear convex sum weights for the ChI of this walk (sorting) are $(0.027, 0.473, 0.45, 0.05)$ respectively. Thus, it is a weighted average of GoogLeNet and ResNet50. This analytic process can be repeated for the other $N! - 1$ walks if desired.




(a)




(b)

Figure 11.6

Learned full ChI per neuron on fold 1 of the UCM dataset. (a) Plot of the $2^4 - 1$ binary encoded FM variables, i.e., for $N = 3$ the order is $(x1, x2, x12, x3, x13, x23, x123, x4, x14, x24, x124, x34, x134, x234, x1234)$. This plot gives us an idea about the agreement/disagreement of variable values across the 21 neurons. If all neurons required the same fusion then each x-axis location would have a single convergent set of circles (FM variable values). However, we can clearly see that each x-axis location (FM variable) has for the most part significant variation (outside the CaffeNet density). (b) Plot of the $4$ neuron Shapley index values across the 21 neurons. Again, this plot shows the variety of values learned. With respect to individual output neurons, some NNs could be eliminated. However, across the 21 neurons, there is no single NN that can be eliminated (we would expect to see approximately all zero values for that Shapley if so).

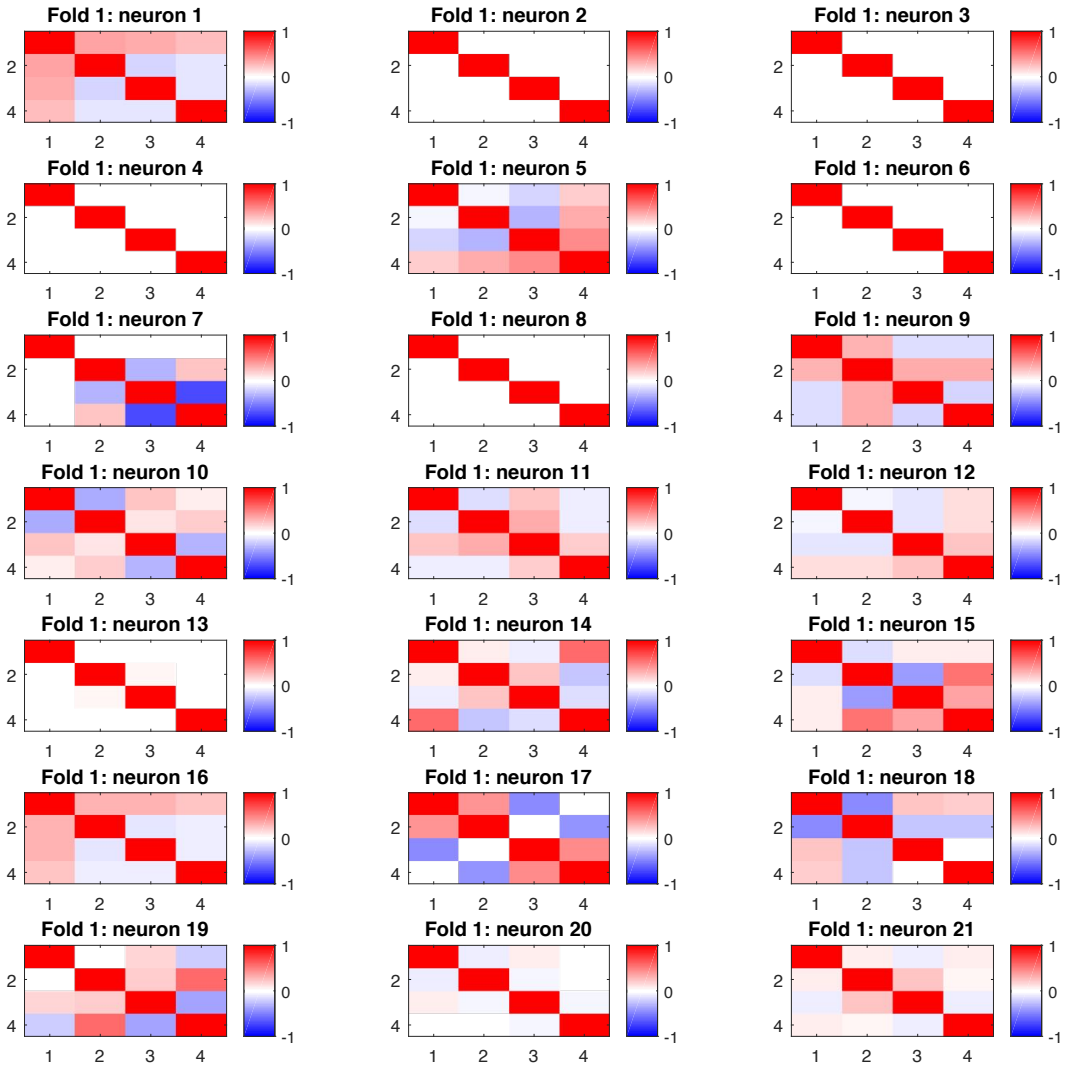



Figure 11.7
Interaction index values for the learned full ChI per neuron on fold 1 of the UCM dataset. Index 1 is CaffeNet, 2 is GoogLeNet, 3 is ResNet50 and 4 is ResNet101. Consider neuron 1. CaffeNet has positive interactions (complementary information) with the other three NNs (0.37, 0.34 and 0.3 respectively). On the other hand, GoogLeNet has negative interaction values (redundancy) with the ResNet NNs (-0.19 and -0.1 respectively). The two ResNet NNs have a negative interaction index of -0.12. Also, in neuron 7, CaffeNet has approximately a zero interaction index with the other NNs (independence), whereas GoogLeNet has a value of -0.29 with ResNet50 and a positive interaction value of 0.22 with ResNet101. Last, ResNet50 and ResNet101 have a large negative interaction index of -0.72 with each other.

| Misclassified Image | Other images from correct class | Predicted Class Exemplar |
|---|---|---|
|  |   |  |
| a | Dense Residential | Mobile Home Park |
|  |   |  |
| b | Intersection | Overpass |
|  |   |  |
| c | Medium Residential | Dense Residential |
|  |   |  |
| d | Golf Course | Forest |
|  |   |  |
| e | Dense Residential | Medium Residential |

Figure 11.8
Five images *missed* by the fusion framework; a) dense residential misclassified as mobile home park, b) (incorrectly labeled) intersection misclassified as overpass (correct label), c) medium residential misclassified as dense residentail, d) (incorrectly labeled) golf course misclassified as forest, and e) dense residential misclassified as medium residential.

Last, Figure 11.6 shows the FM and Shapley values. While it is more-or-less impossible to read individual values in these plots, they show that there is no consensus in values nor importance of DCNNs. Meaning, different output neurons (classes) appears to use these different DCNNs in different ways. Furthermore, Figure 11.7 shows the correspond-

ing interaction index values. These values also reinforce the complex interplay and back-and-forth exchange of complementary, independent and redundant information between DCNNs across output neurons (classes). In total, the combination of analysis of underlying aggregation function, importance of individual DCNNs and their pair-wise interaction behavior help the claim that performance appears to be improving due to diversity in the way these DCNNs operate. This is in line with our intuition about these DCNNs based on the ways their architectures were created.

Last, Figure 11.8 shows example images *missed* by our fusion approach. As the reader can visually verify, these examples are extreme and represent incorrectly labeled or fundamentally ambiguous labels. We would not expect fusion to be able to fix this type of problem. At that, it is hard to say that the DCNNs should have got these, as a human might just as well mistaken them.

Table 11.2
Fusion Results for the RSD dataset

| **Accuracy** | **Method** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Full ChI Per Neuron | Full ChI Shared | SLFM ChI Shared | CaffeNet | GoogLeNet | ResNet50 | Max | Avg | Min |
| **Fold 1** | 0.989 | **0.991** | **0.991** | 0.982 | 0.977 | 0.988 | **0.991** | **0.991** | **0.991** |
| **Fold 2** | 0.992 | 0.984 | 0.992 | 0.978 | **0.994** | 0.989 | 0.987 | 0.992 | 0.987 |
| **Fold 3** | 0.984 | **0.992** | 0.979 | 0.955 | 0.988 | 0.966 | 0.979 | 0.979 | 0.979 |
| **Fold 4** | 0.983 | 0.983 | 0.983 | 0.983 | 0.960 | 0.971 | 0.983 | **0.988** | 0.987 |
| **Fold 5** | 0.998 | **1.00** | **1.00** | 0.977 | 0.994 | 0.994 | **1.00** | **1.00** | **1.00** |

Figure 11.9
Sample image chips from the 19 class RSD benchmark dataset, each 600x600 pixels of various spatial resolution. Classes in left-to-right, top-down order: 1 *airport*, 2 *beach*, 3 *bridge*, 4 *commercial area*, 5 *desert*, 6 *farmland*, 7 *football field*, 8 *forest*, 9 *industrial area*, 10 *meadow*, 11 *mountain*, 12 *park*, 13 *parking lot*, 14 *pond*, 15 *port*, 16 *railway station*, 17 *residential area*, 18 *river*, and 19 *viaduct*. In subsection 11.5.2, neuron indices are used instead of text descriptions for sake of compactness.

### 11.5.2 WHU-RS19 (RSD) dataset

The WHU-RS19 (RSD) dataset is composed of 600x600 pixel, JPEG compressed images [51]. This class includes 19 classes, and approximately 50 chips per class. This imagery was screen scrapped from Google Earth, and therefore they are of variable spatial resolutions. Figure 11.9 shows exemplar image chips from each class of the RSD benchmark dataset. Similar to the UCM dataset, this dataset is a mixture of landcover and objects

within the image chips. Table 11.2 shows the result of fusion, Figure 11.10 are the indices of introspection, Figure 11.11 are example lattices, Figure 11.12 are the FM and the Shapley indices and Figure 11.13 are example interaction indices. Overall, we see the same general trend (as the UCM dataset). Namely, i) aggregation outperforms no aggregation in general and ii) there are challenges with variety (and therefore generalizability) in the RSD data set as well.

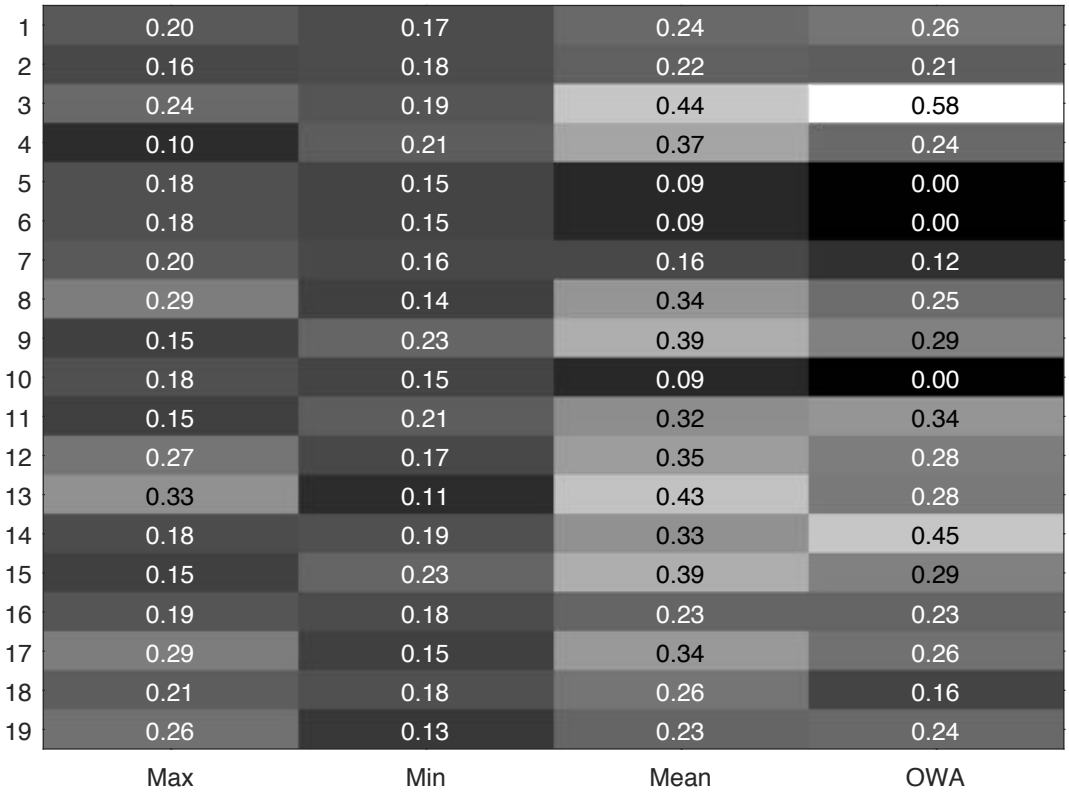
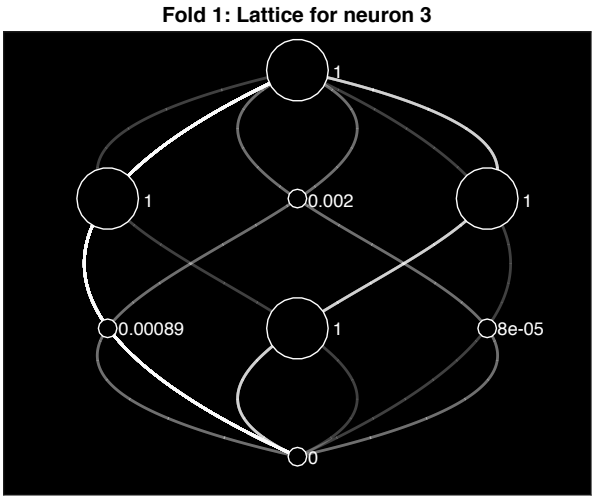
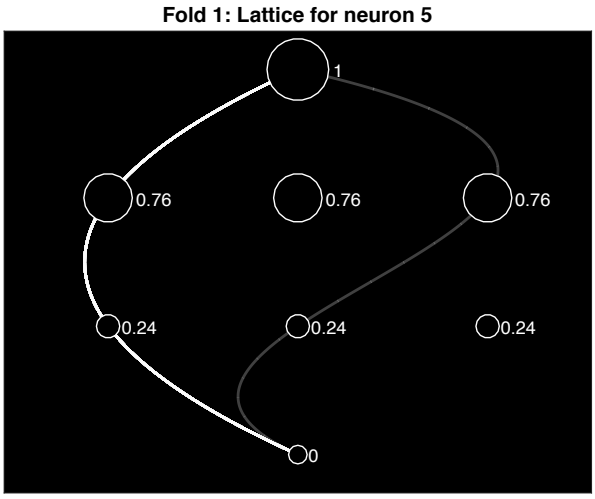



Figure 11.10
Color coded matrix showing the distances obtained using the four reported indices of introspection ($D_1(\mu)$ to $D_4(\mu)$) relative to the learned full ChI per neuron on fold 1 of the RSD dataset. y-axis is the neuron index (see Figure 11.9) and x-axis is the distance measure.



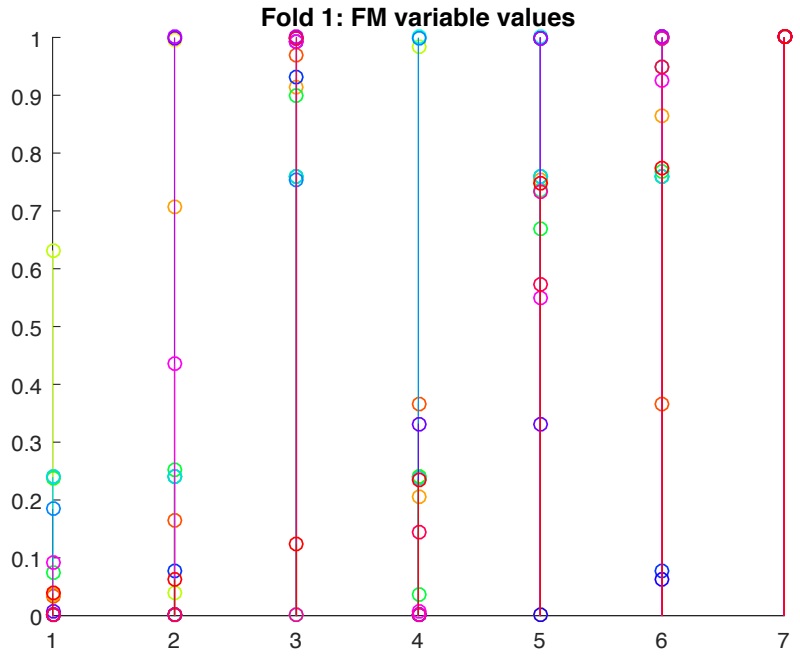

(a)



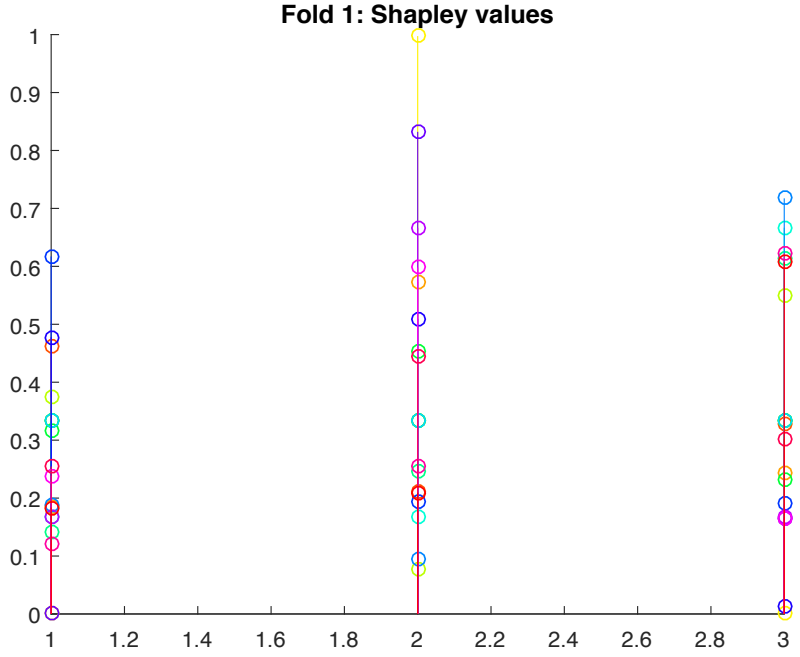

(b)

Figure 11.11

Two example FMs for fold 1 of the RSD dataset. Neuron three is for all intents and purposes a binary FM (see [9] for a formal characterization of binary FMs, the resultant FI and efficient ways of representing and learning such a function). For binary FMs, the Sugeno FI and the ChI are mathematically equivalent [9]. The FI is acting like a "dynamic maximum operator" with respect to FM variables that have a value one–or conversely a "dynamic minimum" with respect to zero valued FM variables. For example, if $h(\{x_1\}) \geq h(\{x_2\}) \geq h(\{x_3\})$ (aka CaffeNet is more confident than GoogLeNet followed by ResNet) then we take the output of GoogLeNet. However, if $h(\{x_2\})$ (GoogLeNet) is ever the most confident then we take its input. This line of reasoning can be followed to get similar stories for the other $N! - 2$ walks. Another interesting observation of neuron 3, versus neuron 5, is a slightly more diverse visitation (walk) pattern.




(a)




(b)

Figure 11.12

Binary encoded $2^3 - 1$ FM variables and the 3 Shapley index values for the nineteen output neurons in the RSD dataset. As demonstrated in the UCM dataset, great variability exists in FM variable and Shapley values for these nineteen output neurons.

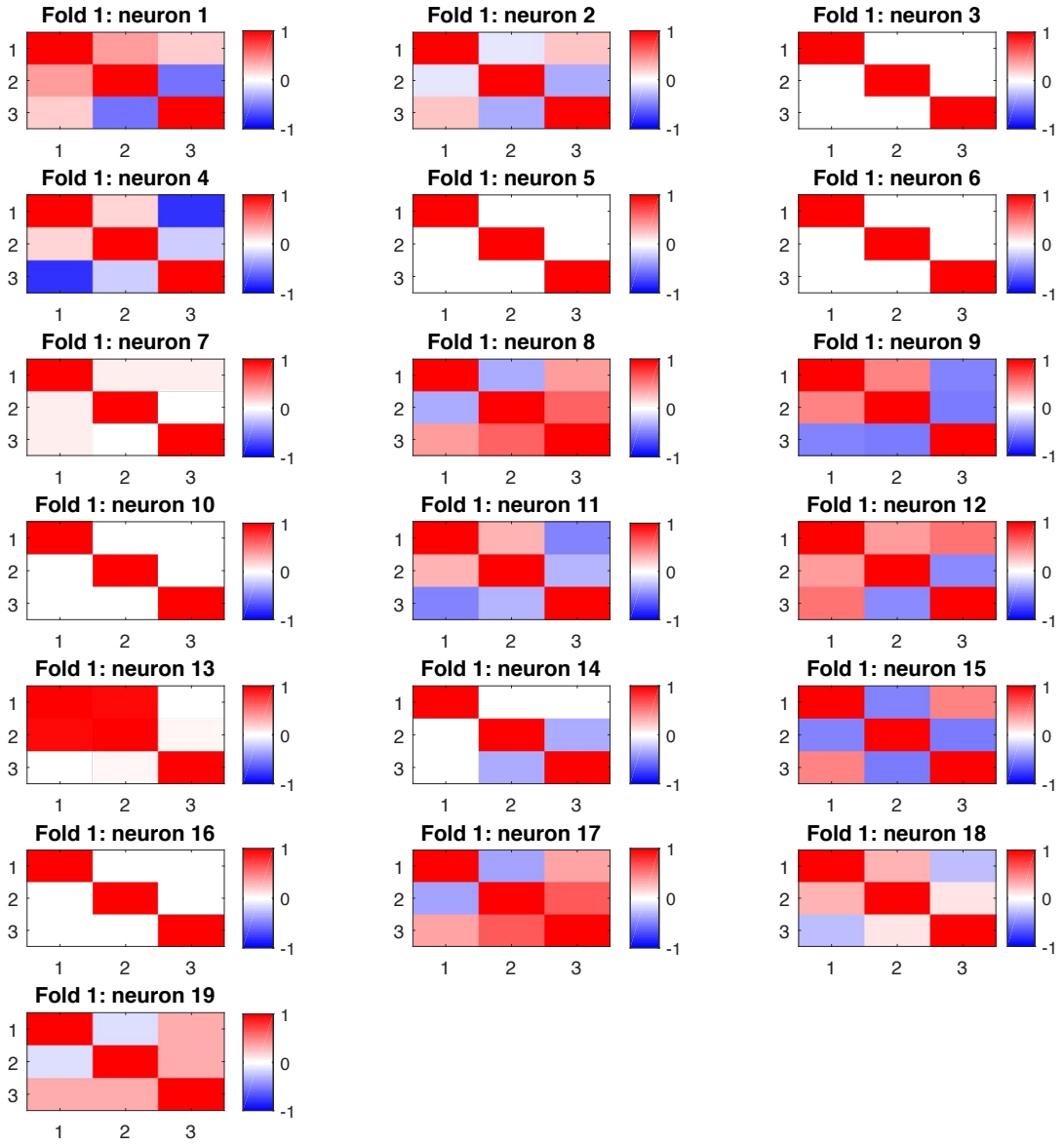



Figure 11.13
Interaction index values for nineteen outputs in the RSD dataset. Index 1 is CaffeNet, 2 is GoogLeNet and 3 is ResNet50.

## 11.6 Conclusion and Future Work

In summary, this chapter outlined a data-driven method for optimizing Choquet integral-based fusion of heterogeneous deep convolutional neural networks for pattern recognition in remotely sensed data. To the best of our knowledge, no one has previously learned

the full fuzzy Choquet integral for fusing neural networks, just density-based fuzzy measures. This chapter brought together state-of-the-art advancements in two important parts of computational intelligence; fuzzy set theory and neural networks. Specifically, CaffeNet, GoogLeNet, ResNet50 and ResNet101 were fused at the per-output-neuron and with respect to a single "shared weight" solution. In a strive for explainable AI, versus black box solutions, different indices of introspection of the Choquet integral and information theoretic indices of the fuzzy measure were highlighted for analysis of the final deep learning solution. These indices showed us that there does appear to be diversity in these different heterogeneous DCNNs. Two benchmark remote sensing datasets were used, UCM and RSD, and our fused results showed improvement over the individual deep learners. However, this data set and DCNNs were saturated and therefore limited data (both volume and variety) existed for training fusion. Last, analysis of mislabeled imagery from our fusion revealed incorrectly labeled data and ambiguous image chips that would likely lead to a human mislabeling imagery.

While encouraging, more research (theory and application) is needed. In future work, we will migrate our Choquet integral solution into a strictly neural representation for optimization and speed. Furthermore, we will move away from DIDO and explore fusion neurons at various layers in the network. We will also investigate what types of neural inputs should be fed to DIDO fusion; e.g., combinations of deep and shallow, different convolution map scales, etc. Future work will also include simultaneously learning the DCNNs and our fusion operators (they are learned independently herein). Last, we will look to use our explainable AI methods to make improvements to the fusion and DCNNs,

manually as well as possibly using them directly computationally to promote diversity and/or aid in the design of our networks.

# CHAPTER XII

# AN EFFICIENT EVOLUTIONARY ALGORITHM FOR OPTIMIZATION OF THE CHOQUET INTEGRAL

## 12.1 Introduction

Data and information fusion, hereafter referred to as data fusion unless there is a reason to differentiate, is an essential part of nearly all state-of-the-art and emerging technologies. For example, self driving cars work on the basis of exploiting sensors like lidar and radar to determine safe navigation in complex and dynamic environments. Similarly, computer vision operates on the basis of extracting and exploiting a diverse set of features for tasks such as object detection. In a geospatial context, remote sensing systems frequently require the intelligent combining of human, sensor (e.g., multi and hyperspectral, synthetic aperture radar, RGB, near infrared, etc.), and algorithm outputs for tasks like object detection, land classification and earth observations, to name a few. These are just a few examples that emphasize the necessity of studying fusion for problems demanding the transformation of data to decisions.

In general, the term "fusion" is too generic; meaning the word has too much variation across domains and users. Herein, we narrow our focus and discuss a piece of the fusion puzzle called aggregation. Let $X = \{x_1, x_2, ..., x_N\}$ be $N$ sources of data, let $z(\{x_i\})$ be the input from source $i$, and let $\mathbf{z} = (z(\{x_1\}), ..., z(\{x_N\}))^t$ be a vector of inputs. An

aggregation operator is a mapping of data-to-data, $f(\mathbf{z}) = \mathbf{y}$. Typically, $\mathbf{y}$ is not multi-dimensional but real-valued, making the goal one of summarization.

Herein, we focus on the *Choquet integral* (ChI), a powerful aggregation operator. The ChI performs nonlinear integration of $\mathbf{z}$ with respect to a *fuzzy measure* (FM). In many applications the FM is learned from available training data via solving an underlying optimization problem. While classical algorithms like gradient descent [121], sub-gradient descent, and interior point can solve convex problems (e.g., quadratic programming [76], linear programming [27, 26], and regularization based ChI [12]), they are not a good fit for non-convex objective functions. To find a solution to challenging and non-analytically solvable optimization tasks, Holland introduced a heuristic search method, the *genetic algorithm* (GA) [93, 36]. Herein, we propose a new efficient GA to solve the task of learning the highly inequality constrained (due to the FM) ChI. However, without loss of generality, our proposed operators can be applied to other evolutionary algorithms (EAs) and problems (beyond the ChI) when/if desired. In particular, the FM is defined over every possible subset of sources, which results in $2^N$ variables. By definition, a FM must satisfy the monotonicity property, which gives rise to a massive number of constraints, $N(2^{N-1} - 1)$. For example, a problem with $10$ inputs has $1,024$ variables and $5,110$ constraints.

Constraint-based optimization with a GA has been addressed in a few indirect ways to date. For example, in [95, 138, 205, 116] any candidate solution that violates a constraint is discarded. Beyond the obvious inefficiency of this approach, a problem is that some of these "invalid" solutions may actually reside in close proximity to a local or global optimum. Second, this approach is not scalable, meaning it suffers from solution starvation as

the number of inputs and thus constraints increase. Another constraint handling approach [53, 47] is to penalize the violating solutions to some degree so that they play a lesser role in the search process. However, an obvious issue with this approach is how do we identify a suitable penalty function and since solutions returned by the process may violate constraints, how is a valid solution obtained? The point is, prior direct (relating to the ChI) and indirect (GA optimization in general) work exists.

Another way to address constrained optimization problems is via cultural algorithms (CAs). A CA can incorporate the knowledge, prior or acquired during the optimization process, to guide the search process. In [115], Jin and Reynolds proposed a CA to solve a nonlinearly constrained optimization problem. The idea is to partition the problem domain into small regions or "cells", and to label the cells as feasible, non-feasible, and unknown, with the knowledge acquired in the search process. As the algorithm progresses, it narrows the search space by removing the undesirable parts and it directs its search toward the region where the solution is most likely to reside. Obviously, this algorithm is highly computational intensive and storage demanding.

Recently, a few have developed customized GAs to address the ChI. In [55], *multiple instance learning based ChI GA* (MICIGA) was put forth. MICIGA identifies the admissible range for each FM variable that does not violate the constraints and it allows the FM to change within that range. The amount of change, either "small" or "large", is stochastically determined by comparing a randomly generated value from a uniform distribution with a user specified threshold. Due to the absence of one of the main GA operators, namely crossover, the quality of the outcome from MICIGA depends primarily on random

exploration and sampling of the initial population from the search space. Furthermore, as random changes occur to each FM variable at each iteration, it is hard to determine when MICIGA converges to a solution. In a separate work [2], the ChI was used for ontology matching. This GA identifies valid candidate pairs. However, as the number of inputs, and thus inequalities increase, few (if any) pairs are available and a condition known as starvation can occur. Furthermore, in [11] the authors explicitly search for violations. If any violation is found, then a bottom-up correction is applied by searching for the closest valid solution. As a result, search is not entirely "smooth" and as the number of inputs increase, more-or-less every solution is constantly being subjected to correction. Last, in [131] Mago and Modave proposed a GA for the 2-additive ChI for MCDM. In particular, a 2-additive ChI restricts interaction to at most two sources, and as a result there are drastically fewer constraints to handle. However, their savings come at the expense of a trade off in representation.

In summary, none of the discussed algorithms offer an efficient way to solve the problem that we are interested in: optimizing the ChI. As a result, herein we propose an efficient algorithm, which we refer to as *efficient ChI GA* (ECGA). The proposed method introduces two new evolutionary operators that naturally enforce the monotonicity and boundary conditions of the FM. These operators can be used to create an offspring from parents during crossover as well as in mutation. Aided by these monotonic and boundary preserving operations, we also provide an efficient representation of the FM. It can be easily shown that the set of FMs is a convex set, which allows expressing a FM as a linear convex sum of the vertices of the FM convex polyhedron. However, the actual number of vertices of this

convex set is exponential to $N$, making it computationally expensive if we wish to generate new points by enumerating vertices. To combat this, we provide a new computationally feasible representation of a FM with only $N$ vertices of the FM set, which we refer to as a minimal set. This representation enables us to generate random points in the feasible solution space as well as prove that any point in the solution space can be reached via re-combination of points. We use three monotonic operators, linear sum, product and *ordered weighted average* (OWA), to create three offspring, which compete against siblings and parents to survive to the next generation. Unlike [55], which uses small scale mutation in the feasible interval, our mutation can make changes beyond the interval bounds, thus providing more degrees of freedom in exploring the solution space.

The remainder of this article is organized as follows. Section II provides an overview of the FM/ChI. Section III details the operations that preserve the FM properties and we introduce the new FM representation. Section IV is the overall GA and experiments are discussed in Section V.

## 12.2 The Fuzzy Measure and Choquet Integral

Non-additive fuzzy integrals, such as the ChI, are defined with respect to a non-additive FM. The ChI provides a flexible way to fuse multiple inputs in a nonlinear manner. The FM, $g : 2^X \rightarrow \mathbb{R}^+$, is a function with the following two properties; (i) (boundary condition) $g(\emptyset) = 0$, and (ii) (monotonicity) if $A, B \subseteq X$, and $A \subset B$, then $g(A) \leq g(B)$.

Without loss of generality, sometimes a normality condition is imposed upon the FM for simplicity and convenience such that $g(X) = 1$. Throughout this paper, we consider

this condition, which results in $2^N - 2$ variables since two FM variables, $g(\emptyset)$ and $g(X)$ are constants due to boundary and normality conditions. Next, we discuss the concept of a partially ordered set relative to the FM. This is critical to our representation and subsequent optimization.

**Definition 1 (Partially ordered set)**

*Let $A = \{a_i\}, i = 1, 2, ..., r$ be an arbitrary set and $R$ be a reflexive, antisymmetric, and transitive binary relation on $A$. Then $\mathbf{P} = (A, R)$ is called a partially ordered set (poset), where $A$ is called the ground set and $R$ is the partial order [188].*

It is trivial to show that a FM is a poset (see Def. 3), where the monotonicity constraints characterize the inequality relations among its variables.

**Definition 2 (FM poset)**

*Let $F = \{g(\{x_1\}), \ldots, g(X)\}$ be a lexicographically ordered FM. According to the definition of a FM and Def. 1, $F$ is a poset.*

**Definition 3 (Monotonic poset)**

*A monotonic poset is $g' : 2^X \rightarrow \mathbb{R}$ that preserves the FM monotonicity property and may or may not hold the boundary and normality conditions. As such, let $F' = \{g'(\{x_1\}), ..., g'(X)\}$ be a lexicograhically ordered poset, where $g'$ lies on the real-valued line versus $[0, 1]$.*

Let $\mathcal{F} = \{F\}$ be the set of all FM posets and $\mathcal{F}' = \{F'\}$ be the set of all monotonic posets. As $\mathcal{F}'$ encompasses all FM posets, $\mathcal{F} \subset \mathcal{F}'$.

**Definition 4 (Anti-monotonic poset)**

*An anti-mononotic poset, $g'' : 2^X \rightarrow \mathbb{R}$ is a poset, where $g''(x)$ is monontonically non-increasing (versus mononotonically non-decreasing), i.e., if $A, B \subseteq X$, and $A \subset B$, then $g''(A) \geq g''(B)$.*

An anti-monotonic poset can easily be converted to a monotoinicy poset and vice versa by simply multiplying the elements in the set by $-1$, which flips the inequality relations.

**Definition 5 (Choquet integral)**

*The ChI of an observation $\mathbf{z} = (z_1, z_2, \ldots, z_N)^t$, where $z_i = z(\{x_i\})$, on $X$ with respect to FM $g$ is*

$$C_g(\mathbf{z}) = \sum_{j=1}^{N} (z_{\pi(j)} - z_{\pi(j-1)}) g(A_{\pi(j)}),$$

*for $A_{\pi(j)} = \{x_{\pi(j)}, \ldots, x_{\pi(N)}\}$, permutation $\pi$ such that $z_{\pi(1)} \geq z_{\pi(2)} \geq \ldots \geq z_{\pi(N)}$, and $z_{\pi(0)} = 0$ [12].*

## 12.3 Foundation for Evolutionary Operators

In this section, the mathematics that facilitate the efficient optimization of the ChI are discussed. These are not the final mutation and crossover operations (provided in Section 12.4), rather they are the numeric operations used in the EA operators. This section focuses on i) unary and multi-ary monotonic operations, ii) boundary preservation and iii) an efficient representation of a FM.

### 12.3.1 FM Property Preserving Operations

**Proposition 2** ***(Unary monotonic)*** *A function $f : \mathcal{F}' \rightarrow \mathcal{F}'$ is a unary monotonic operation if it is monontonically non-decreasing, i.e., $\frac{df}{dt} \geq 0, t \in \Re$.*

Proof: Let $\mathcal{F}' = \{\{s_1, s_2\} : \forall s_1, s_2 \in \Re \text{ and } s_1 \geq s_2\}$ be a set of posets with partial order relation $s_1 \geq s_2$ and $F' = \{a, b\}, F' \in \mathcal{F}'$. The function $f$ operates elementwise on $F'$ and maps it to $G = \{f(a), f(b)\}$. In order for $G$ to be in $\mathcal{F}'$, $f(a)$ and $f(b)$ need to satisfy the partial order relations $f(a) \geq f(b)$, meaning that $f$ has to be monotonicially non-decreasing, i.e., $f(a) \geq f(b)$ when $a \geq b$. This condition can be written using the basic definition of gradient at point $b$ as $\frac{df}{dt}\big|_{t=b} = \frac{f(a)-f(b)}{a-b} \geq 0$. This concludes our proof.

■

Some examples of unary operations are: (i) multiplication by a constant: $f(t) = wt,\ w \in \mathbb{R}^+$ and (ii) addition or subtraction by constant: $f(t) = t \pm c$. This definition can easily be extended to multi-ary operations that combine a number of FM posets to produce a monotonic poset.

**Proposition 3** ***(Multi-ary monotonic)*** *A multi-ary monotonic operation $f : \mathcal{F}' \times \cdots \times \mathcal{F}' \times \cdots \times \mathcal{F}' \rightarrow \mathcal{F}'$, is a $n$-variate monontonically non-decreasing function $f(t_1, t_2, \ldots, t_i, \ldots, t_n)$, i.e., $\frac{\partial f}{\partial t_i} \geq 0, t_i \in \Re$.*

Let $\mathcal{F}' = \{\{s_1, s_2\} : \forall s_1, s_2 \in \Re \text{ and } s_1 \geq s_2\}$ be a set of posets with partial order relation $s_1 \geq s_2$ and $F_1' = \{a, b\}$, $F_2' = \{c, d\}, F_1', F_2' \in \mathcal{F}'$. Then a 2-variate non-decreasing function $f$ can combine $F_1'$ and $F_2'$ to create a new poset $F_3' = \{f(a, c), f(b, d)\}, f(a, c) \geq f(b, d)$ that is also in $\mathcal{F}'$. This proposition can easily be proved for arbitrary $N$ posets using the same procedure described in Proposition 2.

Examples of monotonic operators with two (or more) variables include the (i) linear conic sum ($f(t_1, t_2) = w_1 t_1 + w_2 t_2,\ w_1, w_2 \in \mathbb{R}^+$), (ii) monomial ($f(t_1, t_2) = t_1^{w_1} t_2^{w_2},\ w_1, w_2 \in \mathbb{R}^+$) (iii) numerous t-norms and t-conorms, and (v) the OWA.

Operations, such as the linear convex sum (a special case of linear conic sum where the sum of weights are equal to one), most t-norm and t-conorm's, and OWAs map the output into $[0, 1]$; thus they also satisfy the boundary condition and therefore directly map to a valid FM. However, many other operators do not, and as such we need to apply boundary preserving operations. Next, we propose an operation that enforces the boundary and normality conditions on $F'$, and thus transforms the monotonic poset, $F'$ to a valid FM poset, $F$.

**Definition 6 (Boundary fixing operation)**

*Let the function for the boundary fixing operation be defined as*

$$f(t) = \begin{cases} 0 & \textit{if } t = g'(\emptyset) \\ 1 & \textit{if } t = g'(X) \\ \max(\min(1, t), 0) & \textit{else} \end{cases}.$$

It can easily be shown that the above example operators preserve all the FM properties; however, we demonstrate it for one case, the linear convex sum in Section 12.3.2.

### 12.3.2 Efficient FM Representation

In order to facilitate a geometric interpretation and ultimately an efficient representation of $F$, we regard elements in $F$ as a point $(\mathbf{p})$ in a multi-dimensional space. First, we show that $F$ is a convex set, meaning any point in this set can be represented as a linear convex

sum of its vertices. Second, we show an efficient way to represent a FM in terms of drastically fewer points (versus all of its verticies).

**Proposition 4** ***(FM as a convex set)*** *$\mathcal{F}$, the set of all FMs on X is a convex set. That is, the convex combination of any number of points in $\mathcal{F}$ also lies in $\mathcal{F}$, i.e., $\sum_i^r w_i \mathbf{p}_i \in F$ for $r \in \mathbb{N}, \mathbf{p}_i \in \mathcal{F},\ w_i \geq 0$ and $\sum_i^r w_i = 1$.*

Proof: According to Proposition 3, linear sum preserves monotonicity. As such, the linear convex sum of FM in a multi-dimensional space, i.e., $\sum_i^r w_i \mathbf{p}_i$ is also monotonic. Furthermore, linear convex sum yields zero when all inputs are zeros, one when all inputs are ones and its output is bounded within $[0, 1]$ if the inputs range is $[0, 1]$. Therefore, linear convex sum preserves the boundary conditions. Since $\sum_i^r w_i \mathbf{p}_i$, where $w_i \geq 0$ and $\sum_i^r w_i = 1$, has all the properties of a FM, this concludes our proof. ■

Since $\mathcal{F}$ is a convex set it forms a convex polyhedron. The vertices of this polyhedron can be obtained by solving the inequality and equality constraints. An example is provided next. However, we remark that Avis and Fukuda provided an efficient algorithm to enumerate the vertices from constraints [19].

**Example 6.** In this example we illustrate the aforementioned FM representation for $N = 3$. Let the FM be $\mathbf{g} = \{g_1,\ g_2,\ g_3,\ g_{12},\ g_{13},\ g_{23},\ g_{123}\}$, which is shorthand notation for readability, i.e., $g_{12}$ stands for $g(\{x_1, x_2\})$. The boundary, normality, and monotonicity properties are:

$$g_{\emptyset} = 0 \qquad (12.10)$$

$$g_1 \geq 0 \qquad (12.1) \qquad g_{12} \leq 1 \qquad (12.11)$$

$$g_2 \geq 0 \qquad (12.2) \qquad g_{23} \leq 1 \qquad (12.12)$$

$$g_3 \geq 0 \qquad (12.3) \qquad g_{12} \geq g_1 \qquad (12.13)$$

$$g_{12} \geq 0 \qquad (12.4) \qquad g_{12} \geq g_2 \qquad (12.14)$$

$$g_{13} \geq 0 \qquad (12.5) \qquad g_{13} \geq g_1 \qquad (12.15)$$

$$g_{23} \geq 0 \qquad (12.6) \qquad g_{13} \geq g_3 \qquad (12.16)$$

$$g_1 \leq 1 \qquad (12.7) \qquad g_{23} \geq g_2 \qquad (12.17)$$

$$g_2 \leq 1 \qquad (12.8) \qquad g_{23} \geq g_3 \qquad (12.18)$$

$$g_3 \leq 1 \qquad (12.9) \qquad g_{123} = 1.$$

Since the FM values for the empty set and X are constants, we can perform our analysis based on the remaining $2^N - 2$, or $6$, variables. Equations (12.1)-(12.18) determine the feasible solution space of these $6$ variables. Solving these equations analytically, we obtain the vertices of the solution space, a $6$-dimensional convex polyhedron. Table 12.2 lists the vertices along with the corresponding intersecting equations. Furthermore, any point with the convex polyhedron can be represented as the linear convex sum of these vertices, i.e.,

$$\mathbf{p}_i = \sum_i w_i \mathbf{v}_i, w_i \geq 0, \sum_i w_i = 1.$$

Table 12.1
Coefficients for the minimal set for generating vertices

| $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | Resultant vertex |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | $\mathbf{v}_1$ |
| 1 | 0 | 0 | 0 | 0 | 0 | $\mathbf{v}_2 = \mathbf{m}_1$ |
| 0 | 1 | 0 | 0 | 0 | 0 | $\mathbf{v}_3 = \mathbf{m}_2$ |
| 0 | 0 | 1 | 0 | 0 | 0 | $\mathbf{v}_4 = \mathbf{m}_3$ |
| 1 | 1 | 0 | 0 | 0 | 0 | $\mathbf{v}_5$ |
| 1 | 0 | 1 | 0 | 0 | 0 | $\mathbf{v}_6$ |
| 0 | 1 | 1 | 0 | 0 | 0 | $\mathbf{v}_7$ |
| 1 | 1 | 1 | 0 | 0 | 0 | $\mathbf{v}_8$ |
| 0 | 0 | 0 | 1 | 0 | 0 | $\mathbf{v}_9 = \mathbf{m}_4$ |
| 0 | 1 | 0 | 0 | 1 | 0 | $\mathbf{v}_{10} = \mathbf{m}_5$ |
| 0 | 0 | 0 | 0 | 0 | 1 | $\mathbf{v}_{11} = \mathbf{m}_6$ |
| 1 | 0 | 0 | 1 | 0 | 0 | $\mathbf{v}_{12}$ |
| 0 | 1 | 0 | 0 | 1 | 0 | $\mathbf{v}_{13}$ |
| 0 | 0 | 1 | 0 | 0 | 1 | $\mathbf{v}_{14}$ |
| 0 | 0 | 0 | 1 | 1 | 0 | $\mathbf{v}_{15}$ |
| 0 | 0 | 0 | 1 | 0 | 1 | $\mathbf{v}_{16}$ |
| 0 | 0 | 0 | 0 | 1 | 1 | $\mathbf{v}_{17}$ |
| 1 | 1 | 1 | 0 | 0 | 0 | $\mathbf{v}_{18}$ |

**Minimal FM Representation:** Let the set of vertices of the FM convex polyhedron be $V = \{\mathbf{v}_j\}$. Building on Example 1, we now define a minimal set of vertices, $M = \{\mathbf{m}_i\}, i = 1, 2, \ldots, 2^N - 2$. Specifically, vertices in $M$ can only be represented by themselves, i.e., they cannot be expressed as a linear convex sum of the remaining vertices in $V$. There exists exactly one $\mathbf{m}_i \in M$ corresponding to each variable, $g(A), A \in 2^X \setminus X$ such that

$$m_i(B) = \begin{cases} 1 & \text{if } B \supseteq A \\ 0 & \text{else} \end{cases} . \tag{12.19}$$

Any vertex $\mathbf{v} \in V$ in the polyhedron can be expressed in terms of the minimal set using the following equation

$$\mathbf{v} = \min(\mathbf{1}, \sum_{i}^{2^N-2} c_i \mathbf{m}_i), c_i \in \{0, 1\}. \tag{12.20}$$

The minimal set for Example 1 is $M = \{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4, \mathbf{m}_5, \mathbf{m}_6\}$, which correspond to vertices $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$, $\mathbf{v}_9$, $\mathbf{v}_{10}$, and $\mathbf{v}_{11}$ respectively (from Table 12.2). Table 12.1 shows that all of the vertices in $\mathcal{F}$ can be expressed in terms of the minimal set, $M$, with respect to its corresponding coefficients ($c$'s). Even though we show this for $N = 3$, it is easy to generalize to the case of any $N$.

In summary, the significance of Eq. 12.19 is it tells us that we can work with a small set of vertices and exploit it in EA operations like weighted recombination to produce new valid points (FMs). This efficient representation forms the basis of our EA search algorithm.

Table 12.2
Vertices for 3 sources

| Vertex | $g_1$ | $g_2$ | $g_3$ | $g_{12}$ | $g_{13}$ | $g_{23}$ | Intersecting equations |
|---|---|---|---|---|---|---|---|
| $\mathbf{v}_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 12.1∼12.6, 12.13∼12.18 |
| $\mathbf{v}_2$ | 0 | 0 | 0 | 0 | 0 | 1 | 12.1∼12.5, 12.12 |
| $\mathbf{v}_3$ | 0 | 0 | 0 | 0 | 1 | 0 | 12.1∼12.4, 12.6, 12.11 |
| $\mathbf{v}_4$ | 0 | 0 | 0 | 1 | 0 | 0 | 12.1∼12.3, 12.5, 12.6, 12.10 |
| $\mathbf{v}_5$ | 0 | 0 | 0 | 0 | 1 | 1 | 12.1∼12.4, 12.11, 12.12 |
| $\mathbf{v}_6$ | 0 | 0 | 0 | 1 | 0 | 1 | 12.1∼12.3, 12.5, 12.10, 12.12 |
| $\mathbf{v}_7$ | 0 | 0 | 0 | 1 | 1 | 0 | 12.1∼12.3, 12.6, 12.10, 12.11 |
| $\mathbf{v}_8$ | 0 | 0 | 0 | 1 | 1 | 1 | 12.1∼12.3, 12.10∼12.12 |
| $\mathbf{v}_9$ | 1 | 0 | 0 | 1 | 1 | 0 | 12.2, 12.3, 12.6, 12.7, 12.10, 12.11, 12.13, 12.15 |
| $\mathbf{v}_{10}$ | 0 | 1 | 0 | 1 | 0 | 1 | 12.1, 12.3, 12.5, 12.8, 12.10, 12.12, 12.14, 12.17 |
| $\mathbf{v}_{11}$ | 0 | 0 | 1 | 0 | 1 | 1 | 12.1, 12.2, 12.4, 12.9, 12.11, 12.12, 12.16, 12.18 |
| $\mathbf{v}_{12}$ | 1 | 0 | 0 | 1 | 1 | 1 | 12.2, 12.3, 12.7, 12.10∼12.13, 12.15, |
| $\mathbf{v}_{13}$ | 0 | 1 | 0 | 1 | 1 | 1 | 12.1, 12.3, 12.8, 12.10∼12.12, 12.14, 12.17, |
| $\mathbf{v}_{14}$ | 0 | 0 | 1 | 1 | 1 | 1 | 12.1, 12.2, 12.9∼12.12, 12.16, 12.18, |
| $\mathbf{v}_{15}$ | 1 | 1 | 0 | 1 | 1 | 1 | 12.3, 12.7, 12.8, 12.10∼12.15, 12.17 |
| $\mathbf{v}_{16}$ | 1 | 0 | 1 | 1 | 1 | 1 | 12.2, 12.7, 12.9∼12.13, 12.15, 12.16, 12.18 |
| $\mathbf{v}_{17}$ | 0 | 1 | 1 | 1 | 1 | 1 | 12.1, 12.8∼12.12, 12.14, 12.16∼12.18 |
| $\mathbf{v}_{18}$ | 1 | 1 | 1 | 1 | 1 | 1 | 12.7∼12.18 |

## 12.4 Efficient GA for the ChI

The previous sections outlined the mechanics (operators and representation) of our evolutionary optimization approach. In this section, we focus on how to use these ideas for (i) initialization, (ii) selection, (iii) crossover and (iv) mutation in a GA. Again, while we focus on a GA, the operators discussed are equally applicable for other EA-based optimization approaches.

### 12.4.1 Initialization

Our first action is to determine an initial set of candidate solutions. In order to ensure that subsequent operations have free range over the full feasible space, we begin by randomly picking points in a range between 1 to $N$ from $M$ (see Eq. 12.19). Next, we generate $n_p - N$ additional individuals using equation

$$\mathbf{p} = \min(\mathbf{1}, \sum_{j=1}^{n_r} w_j \mathbf{t}_j),\ \mathbf{t}_j \in M \text{ and } 1 \leq n_r \leq N \tag{12.21}$$

where $n_p$ is the population size, $n_r \sim U(1, N)$ is an integer randomly selected from a uniform distribution, $\mathbf{t}_j$ is a point randomly selected from $M$ and $w_j \in U(0, 1)$ is the corresponding weight sampled from a uniform distribution.

**Algorithm 9:** Initialization

```
Input initial population size, n_P
Initialize input population, P, to empty
Generate the minimal set, M = {m_k}, k = 1, 2, ..., N using Eq. 12.19 and add these to P
for i = 1, ..., n_P − N do
    Randomly select the number of vertices, s ~ U(1, N)
    Randomly pick n_r vertices, t_j ∈ M, j = 1, 2, ..., s
    Select w_j ~ U(0, 1), j = 1, 2, ..., s
    Create a new valid FM, p, using Eq. 12.21
    Add p to P
Return P
```

#### 12.4.1.1 Selection

Any GA selection method is applicable here. However, in this article we use non-linear ranking with stochastic selection and elitism (to ensure that that the best solution is not lost from one iteration to the next). Algorithm 15 summarizes our selection process.

#### 12.4.1.2 Crossover

While multiple parents can be recombined to spawn an offspring, we limit our discussion to the case of two herein. Our approach involves a monotonic operation followed by the boundary operation, discussed in Section III. Specifically, in this section we focus on three operators that utilize sum, product and sorting operations. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the parents. The three methods explored for generating a child are the following:

1. linear sum: $\tilde{\mathbf{p}}_1 = c_{11}\mathbf{p}_1 + c_{12}\mathbf{p}_2$,

2. monomial: $\tilde{\mathbf{p}}_2 = \mathbf{p}_1^{c_{21}}\mathbf{p}_2^{c_{22}}$,

3. OWA: $\tilde{\mathbf{p}}_3 = c_3 \max(\mathbf{p}_1, \mathbf{p}_2) + (1 - c_3)\min(\mathbf{p}_1, \mathbf{p}_2)$,

where the coefficients $c_{11}$, $c_{12}$, $c_{21}$, $c_{22}$, and $c_3$ are randomly generated from a uniform distribution, $U(0, 1)$. Figure 12.1 illustrates where the offspring reside in relation to the parents. Algorithm 6 is a formal description of our crossover algorithm.

#### 12.4.1.3 Mutation

Mutation is what truly enables global search for solutions over the feasible space. Algorithm 14 is a formal description of our approach.

**Algorithm 10:** Selection

Input population, $P$

Input number of offsprings to select, $n_o$

Calculate the rank of an individual, $\mathbf{p}_i \in P$ based on its position in the ascending sorting order of fitness scores, $f_r(\mathbf{p}_i)$

Square root the rank, $f_s(\mathbf{p}_i) = \sqrt{f_r(\mathbf{p}_i)}$

Calculate probability of an individual $\mathbf{p}_i$ as $\psi(\mathbf{p}_i) = \frac{f_s(\mathbf{p}_i)}{\sum_i f_s(\mathbf{p}_i)}$

**for** $i = 1, \ldots, n_o$ **do**

    $\lambda_i = 0$

$r \sim U(0, \frac{1}{n_s})$

$sum = 0$

**for** $i = 1, \ldots, n_s$ **do**

    $sum = sum + \psi(\mathbf{p}_i)$

    **while** $sum \leq r$ **do**

        $\lambda_i = \lambda_i + 1$

        $r = r + \frac{1}{n_o}$

Return index of the selected individuals, $\lambda = (\lambda_1, \cdots, \lambda_{n_s})$

**Algorithm 11:** Crossover using recombination of two parents

Input crossover rate, $r_C$

Input the parents, $\mathbf{p}_1$, and $\mathbf{p}_2$

**if** $r \sim U(0, 1) \leq r_C$ **then**

    Randomly select the coefficients, $c_{11}$, $c_{12}$, $c_{21}$, $c_{22}$, and $c_3$ from a uniform distribution

    Calculate the three offsprings, $\tilde{\mathbf{p}}_1 = c_{11}\mathbf{p}_1 + c_{12}\mathbf{p}_2$, $\tilde{\mathbf{p}}_2 = \mathbf{p}_1^{c_{21}}\mathbf{p}_2^{c_{22}}$, and $\tilde{\mathbf{p}}_3 = c_3 \max(\mathbf{p}_1, \mathbf{p}_2) + (1 - c_3)\min(\mathbf{p}_1, \mathbf{p}_2)$

Evaluate the fitness scores of $\mathbf{p}_1$, $\mathbf{p}_2$, $\tilde{\mathbf{p}}_1$, $\tilde{\mathbf{p}}_2$, and $\tilde{\mathbf{p}}_3$ and return the one with the best fitness score.

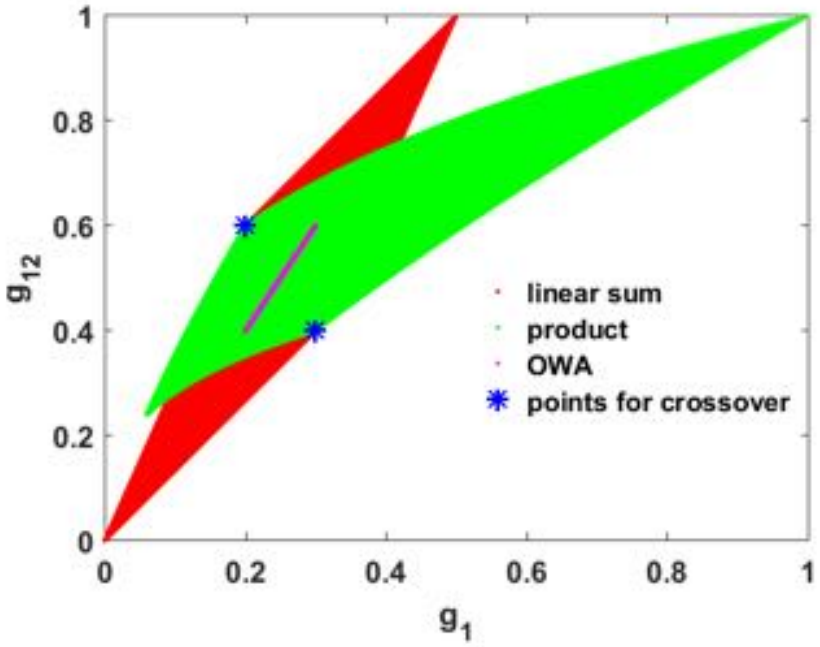



Figure 12.1
Example illustration of crossover for the case of two arbitrary points at $g_1 = (0.2, 0.6)$ and $g_{12} = (0.3, 0.4)$; a subset of the multi-dimensional optimization space. The different sets (represented by color coded points) show the range of the underlying linear sum, product and OWA operators on these two parents.

**Algorithm 12:** Mutation

Input chromosome $\mathbf{p}$

Input mutation point, $i \sim U(1, n)$

Generate a random value, $w \sim U(-1, 1)$

**if** $w \geq 0$ **then**

    Compute upper bound, $ub_{p_i}$ of $p_i$ from its superset variables

    Calculate $p_i = min(ub_{p_i}, p_i + w)$

    Find residual offset, $w_r = max(0, w - ub_{p_i})$

    Combine $\mathbf{p}$ with $i$th vertex in minimal set as $\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r \mathbf{m}_i)$

**else**

    Compute lower bound, $lb_{p_i}$ of $p_i$ from its subset variables

    Calculate $p_i = max(lb_{p_i}, p_i + w)$

    Find residual offset, $w_r = min(0, w - lb_{p_i})$

    Combine $\mathbf{p}$ with $i$th vertex in anti-monotonic set as $\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r \mathbf{q}_i)$

Return $\mathbf{p}$

In Algorithm 14 we start by selecting mutation points, $\mathbf{p}_i$. Next, an offset is selected, $w \in [-1, 1]$, which determines the "direction" of alteration. If this value is positive then we compare it with the feasible upper bound, $ub_{p_i}$, computed from the superset variables of $g_i$ and make the admissible change in $p_i$ within this bound. Last, an residual offset is calculated, $w_r = max(0, w - ub_{p_i})$, and used to combine the individual with the $i$th vertex in $M$ via

$$\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r \mathbf{m}_i).$$

On the other hand, if this shift variable is negative we make an admissible change within the lower bound. and we use the residual offset, $w_r = min(0, w - lb_{p_i})$, to combine with an anti-monotonic point, $\mathbf{q}_i$, to preserve monotonicity. Like the minimal set, we generate an anti-monotonic point, $\mathbf{q}_i \in Q$, for each variable, $g(A), A \subseteq 2^X \setminus X$, as follows

$$q_i(B) = \begin{cases} 1 & \text{if } B \subseteq A \\ \\ 0 & \text{else} \end{cases} . \tag{12.22}$$

We can pre-compute minimal and anti-monotonic sets, $Q$, to increase computational efficiency. Table 12.3 shows the sets, subset and superset variables for the case of Example 1 ($N = 3$). As we can see, the subset and superset variables for each variable can easily be computed from the anti-monotonic set and minimal set respectively. For each variable, $g_i$, we look for variables that are 1s in the corresponding vertex, $\mathbf{m}_i$, which gives the superset (including $g_i$). In order to create strictly superset variables, $g_i$ is excluded. In the same manner, subset variables can be obtained from anti-monotonic set $Q$.

Table 12.3
Minimal set, anti-monotonic set, and subset and superset variables for three inputs

| | | Minimal Set | | | | | | | | Anti-monotonic Minimal Set | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Variables | Vertex | $g_1$ | $g_2$ | $g_3$ | $g_{12}$ | $g_{13}$ | $g_{23}$ | $g_{123}$ | Superset variables | Point | $g_1$ | $g_2$ | $g_3$ | $g_{12}$ | $g_{13}$ | $g_{23}$ | $g_{123}$ | Subset variables |
| $g_1$ | $\mathbf{m}_1$ | 1 | 0 | 0 | 1 | 1 | 0 | 1 | $g_{12}, g_{13}, g_{123}$ | $\mathbf{q}_1$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $\emptyset$ |
| $g_2$ | $\mathbf{m}_2$ | 0 | 1 | 0 | 1 | 0 | 1 | 1 | $g_2, g_{12}, g_{123}$ | $\mathbf{q}_2$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | $\emptyset$ |
| $g_3$ | $\mathbf{m}_3$ | 0 | 0 | 1 | 0 | 1 | 1 | 1 | $g_3, g_{13}, g_{123}$ | $\mathbf{q}_3$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | $\emptyset$ |
| $g_{12}$ | $\mathbf{m}_4$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 | $g_{123}$ | $\mathbf{q}_4$ | 1 | 1 | 0 | 1 | 0 | 0 | 0 | $g_1, g_2$ |
| $g_{13}$ | $\mathbf{m}_5$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | $g_{123}$ | $\mathbf{q}_5$ | 1 | 0 | 1 | 0 | 1 | 0 | 0 | $g_1, g_3$ |
| $g_{23}$ | $\mathbf{m}_6$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | $g_{123}$ | $\mathbf{q}_6$ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | $g_2, g_3$ |

## 12.5 Experiments

To demonstrate the quality of ECGA, we compare it against prior work on several different optimization functions. As there are no benchmark nonlinear ChI optimization problems, we propose experiments such that the location of the global optimum varies across cases. Specifically, we compare ECGA to the GA in [11] and [55], referred to hereafter as FVFMGA and MICIGA respectively. We do not compare to [2], as it suffers extremely from early termination due to starvation of crossover pair identification and it does not scale well at all.

The following experiments are performed. First, we demonstrate ECGA, FVFMGA and MICIGA on a small scale problem; three inputs, thus seven variables. Second, we investigate the impact of increasing the number of variables. Third, we demonstrate scalability versus execution time, which gives us insight into the computational complexity of these approaches. Last, we investigate the impact of the different equations for crossover in ECGA.

In each of our optimization tasks, we consider the following problem,

$$\min_{\mathbf{u}} \; h(C_{\mathbf{u}}) = e(\mathbf{u}),$$

subject to

$$u_i \leq u_j \text{ if } u_i = g(A), u_j = g(B) \text{ and } A \subset B,$$

$$\forall i, j \in \{1, 2, \ldots, 2^N - 1\},$$

$$u_i = 1, \text{ if } u_i = g(X),$$

where $\mathbf{u}$ is a vector representation of the FM variables, $h$ is a function of ChI with respect to $\mathbf{u}$ and $e$ is a function of $\mathbf{u}$. For simplicity and computational efficiency, we can chose to use a binary encoded index for variables in $\mathbf{u}$. This means that for a three input case, indices $1, 3,$ and $4$ corresponds to binary representation of $001, 011,$ and $100$ respectively and, consequently, $u_1 = g(\{x_1\}), u_3 = g(\{x_1, x_2\}), u_4 = g(\{x_3\})$ and so forth.

For the first three functions, we use the sum of residuals

$$e_i = \sum_{j=1}^{2^N-1} (u_j - c_j)^2, i = 1, 2, 3. \tag{12.23}$$

The idea is to compare the candidate solution, $\mathbf{u}$, to an underlying solution (ground truth) FM, $C = [c_1 \; c_2 \; \ldots \; c_{2^N-1}]$.

The first three functions are focused on the ChI. In the fourth data set we use a benchmark data set, the Rastrigin function. This is performed in order to demonstrate that our methods are equally applicable to any multi-variate function with inequality constraints. The conventional Rastrigin function spans $[-5.12, 5.12]$ and has a global minimum at zero. Without loss of generality, we modified the function by scaling and translating it

so the function now spans $[0, 1]$ and the global minimum is now at $0.5$. Thus, the function $e_4$, i.e., modified Rastrigin function, is

$$e_4(\mathbf{u}) = 10 * (2^N - 2) + \sum_{j=1}^{2^N-2} [u_j^2 - 10cos(2\pi u_j)], \tag{12.24}$$

Figure 12.2 shows the surface plot of Eq. 12.24.

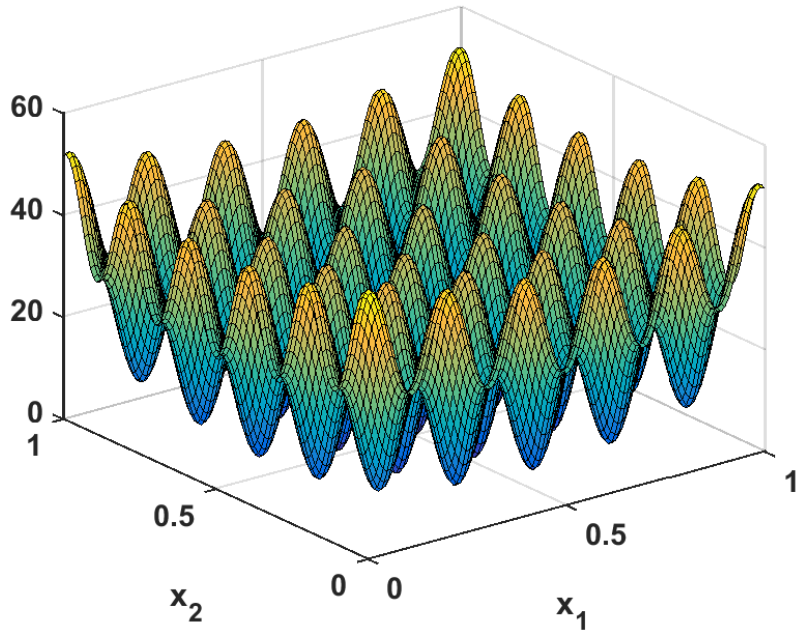



Figure 12.2
Surface plot of the modified Rastrigin function for two variables, which is the fourth data set explored herein. A non-ChI data set was selected to demonstrate the generalizability of our proposed ECGA.

### 12.5.1 Experiment 1: Small Scale Optimization Problem

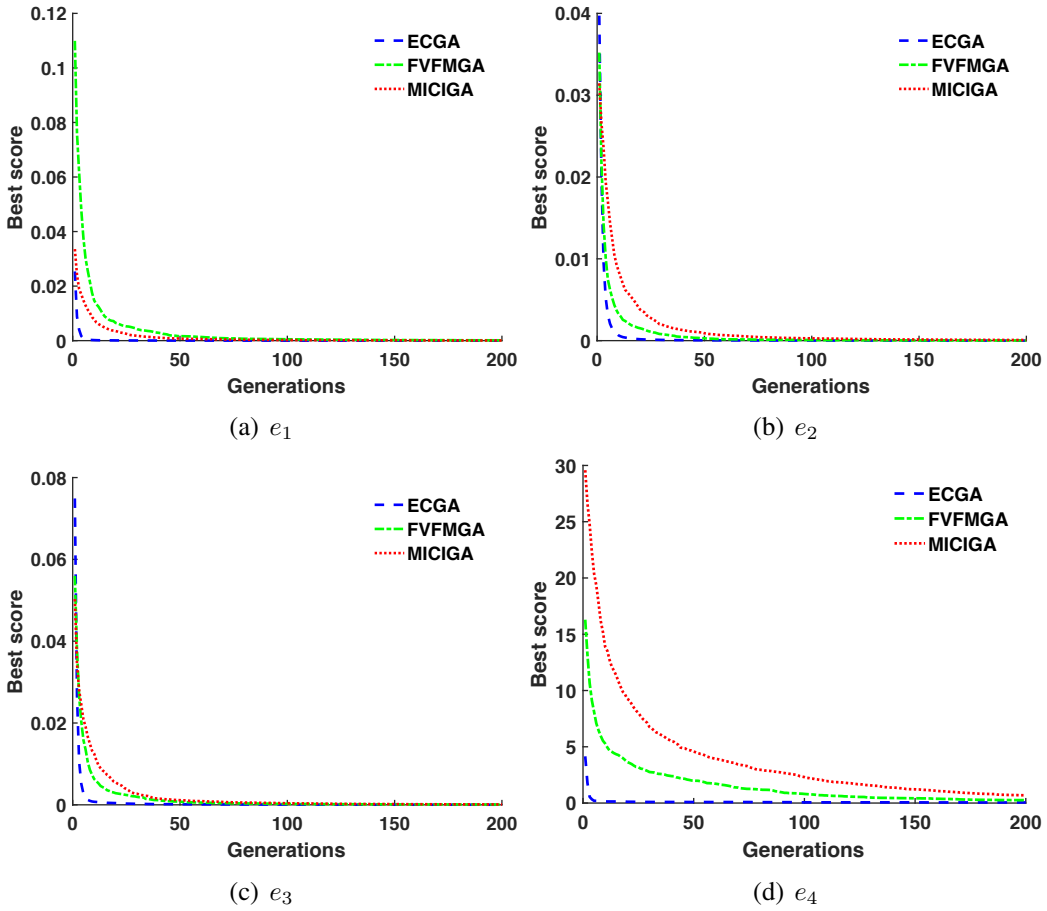
This experiment has N=3, thus seven variables. The function $e_1$, $e_2$ and $e_3$ correspond to OWAs with weights $[0.1, 0.2, 0.7]$, (soft min), $[0.7, 0.2, 0.1]$ (soft max) and $[0.3, 0.4, 0.3]$ (mean like) respectively (see Table 12.4 for coefficients). These three functions were se-

lected in order to span the range of optimistic (union like), pessimistic (intersection like), and expected value (e.g., mean) like. The global optimum for each function is at zero. The fourth function is the Rastrigin function.

Table 12.4
Coefficients for $e_1, e_2,$ and $e_3$ for three inputs

| function | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ |
|---|---|---|---|---|---|---|---|
| $e_1$ | 0.1 | 0.1 | 0.3 | 0.1 | 0.3 | 0.3 | 1 |
| $e_2$ | 0.7 | 0.7 | 0.9 | 0.7 | 0.9 | 0.9 | 1 |
| $e_3$ | 0.3 | 0.3 | 0.7 | 0.3 | 0.7 | 0.7 | 1 |

Binary encoded coefficient index is used.




(a) $e_1$ (b) $e_2$

(c) $e_3$ (d) $e_4$

Figure 12.3
Experiment 1. Comparison of best fitness scores for $N = 3$ for $e_1$, $e_2$, $e_3$, and $e_4$.

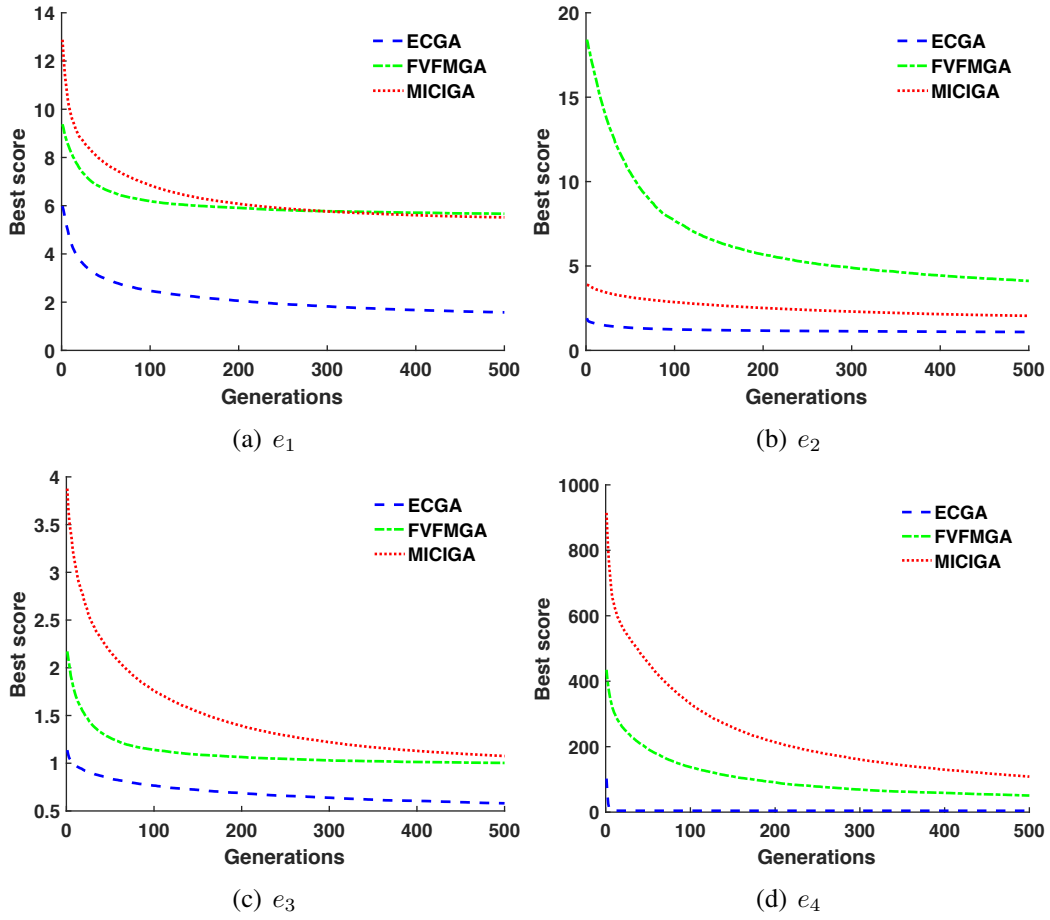
The default parameter settings are as follows. The population size was $100$ and $500$ generations were used. For ECGA and FVFMGA, crossover rates vary from $0.6$ to $0.9$ in step increments of $0.1$ and mutation rates of $0.05, 0.1, 0.2$, and $0.3$ were used. For MICIGA, we used the implementation provided by the authors, which can be found at `https://github.com/GatorSense/MICI`. While MICIGA was designed to solve multiple instance tasks, we adapted it to solve our four functions. MICIGA parameter settings are, small mutation rate from $0.6$ to $0.9$ in step sizes of $0.1$, variance around sample mean of $0.05,\ 0.1,\ 0.2$, and $0.3$. Note MICIGA does not have crossover. Instead it performs small scale mutation. Each experiment was repeated $50$ times and for each method, the average score for the best performer is reported. The best fitness score is used as the evaluation metric.

Figure 12.3 shows the results of Experiment 1. We can see that all three methods achieve the same best fitness score at some point (number of generations). This should be expected for a simple problem. However, ECGA outperforms the other algorithms in all four problems with respect to the number of generations needed to converge. Of the three methods, MICIGA has the worst results, which might be expected as it involves only mutation (taking more generations to get to an answer).

### 12.5.2 Experiment 2: Scalability

Experiment 2 has $N = 6, 63$ variables and $186$ monotonicity constraints. The three error functions had coefficients generated randomly from a uniform distribution with ranges $[0, 1]$, $[0, 0.5]$ and $[0.5, 1]$ respectively. A population size of $200$ and $500$ generations was

ran. The method specific parameter settings are the same as Experiment 1. Figure 12.4 shows the result.




(a) $e_1$ (b) $e_2$

(c) $e_3$ (d) $e_4$

Figure 12.4
Experiment 2. Comparison of best fitness scores for $N = 6$ for $e_1$, $e_2$, $e_3$, and $e_4$.

Figure 12.4 shows the effectiveness of ECGA. Specifically, none of the other methods achieved the same quality of solution. On $e_4$, FVFMGA attained a solution as good as ECGA, but it took approximately $350$ iterations versus $10$. Also, if we look at the best score from the initial population, ECGA has the lowest score for all four error functions. This result is evidence that the initial population of ECGA is likely more distributed over

the solution space. In summary, Experiment 2 demonstrates that our proposed method is suitable for larger scale problems, compared to prior work.

### 12.5.3 Experiment 3: Computational Performance

Our experiments were run on a Windows PC with an Intel Xeon CPU EP-2650 at 2GHz clock speed and 64GB of RAM. All parameters were kept the same for the $N = 3$ and $N = 6$ cases, except for population size. Each experiment was repeated for $50$ times and the average running time for each combination of parameters (e.g., crossover rate = $0.6$ and mutation rate=$0.05$ for ECGA and FVFMGA) was recorded. Table 12.5 lists the minimum, mean, maximum, and the best running time. Here, the best corresponds to the scenario that yields the best results. The number of variables increased by $9$ times from $N = 3$ to $6$ while running time for ECGA, FMFMGA, and MICIGA increased by $3.47$, $8.65$, and $11.91$ respectively. While FVFMGA has the lowest execution time for $N = 3$, it spends on average $1.6$ times more time than ECGA for $N = 6$. The reason is, FVFMGA checks each monotonicity relation for each variable, and there are $N(2^N - 1)$ such checks to make. This causes an extreme computational burden for FVFMGA as $N$ grows. MICIGA has the maximum execution time due to the mutation operation of a large number of variables (60% to 80% of the total variables go through small scale mutation each iteration), which involves calculating the lower and upper bounds and assigning a random value. In summary, we conclude that our method scales well to problem size ($N$).

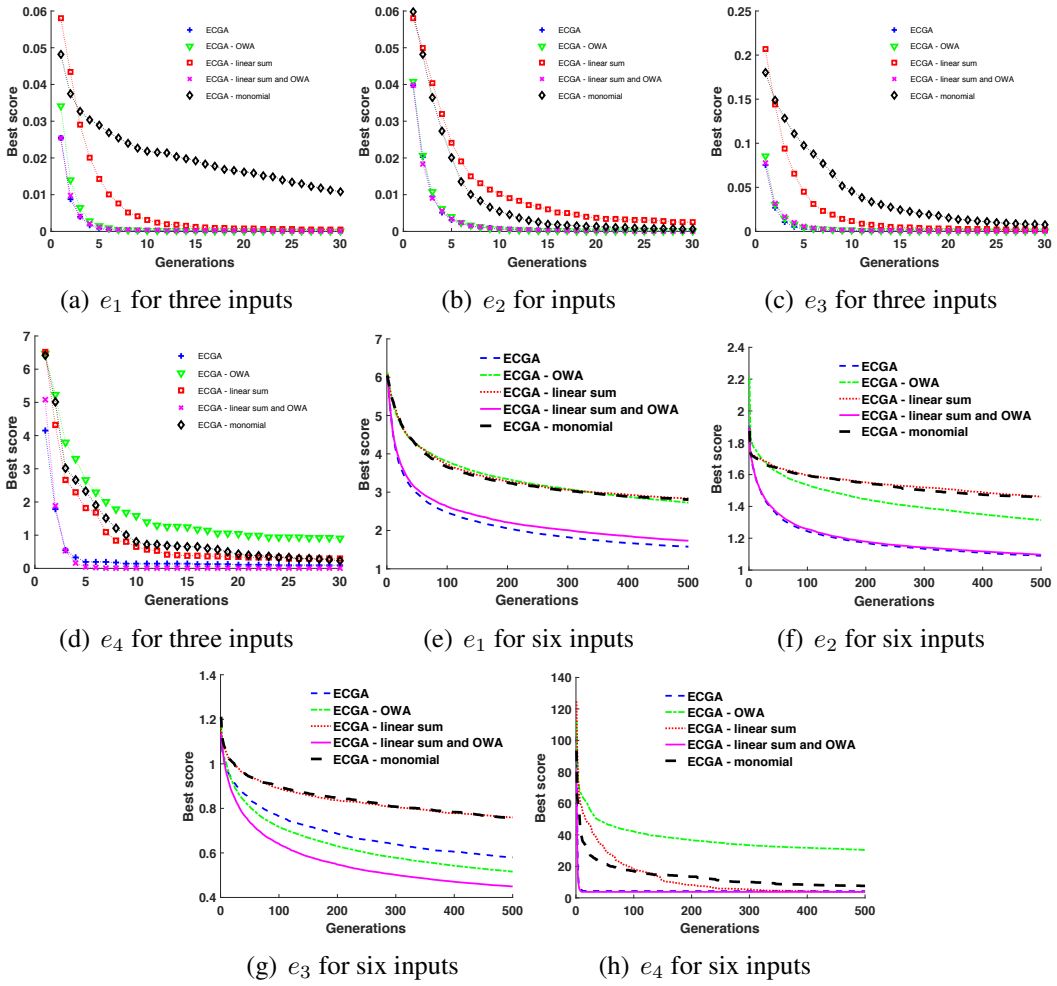
Table 12.5
Execution time in seconds

| | Method | 3 inputs | | | | | 6 inputs | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $e_1$ | $e_2$ | $e_3$ | $e_4$ | Average | $e_1$ | $e_2$ | $e_3$ | $e_4$ | Average |
| Minimum | ECGA | 1.87 | 1.83 | 1.81 | 2.54 | 2.01 | 6.17 | 6.07 | 6.02 | 7.36 | **6.41** |
| | FVFMGA | 1.35 | 1.33 | 1.42 | 1.64 | **1.44** | 12.92 | 12.86 | 12.78 | 12.96 | 12.88 |
| | MICIGA | 5.47 | 5.49 | 5.53 | 5.50 | 5.50 | 52.17 | 44.35 | 48.60 | 51.26 | 49.09 |
| Mean | ECGA | 2.25 | 2.16 | 2.11 | 3.00 | 2.38 | 7.41 | 7.69 | 7.65 | 10.23 | **8.25** |
| | FVFMGA | 1.43 | 1.44 | 1.53 | 1.77 | **1.54** | 13.49 | 13.20 | 13.04 | 13.65 | 13.34 |
| | MICIGA | 6.44 | 6.52 | 6.45 | 6.92 | 6.58 | 82.45 | 71.88 | 74.18 | 85.00 | 78.38 |
| Maximum | ECGA | 3.38 | 2.58 | 2.59 | 3.67 | 3.05 | 10.09 | 10.25 | 10.32 | 12.75 | **10.85** |
| | FVFMGA | 1.71 | 2.16 | 2.19 | 2.10 | **2.04** | 14.97 | 14.64 | 15.35 | 15.14 | 15.02 |
| | MICIGA | 8.24 | 7.77 | 8.07 | 8.69 | 8.19 | 122.40 | 109.69 | 103.43 | 125.05 | 115.14 |
| Best | ECGA | 2.36 | 2.09 | 2.33 | 3.12 | 2.47 | 6.17 | 6.07 | 7.01 | 10.95 | **7.55** |
| | FVFMGA | 1.37 | 1.42 | 1.43 | 1.66 | **1.47** | 13.62 | 12.86 | 12.81 | 12.96 | 13.06 |
| | MICIGA | 7.14 | 7.31 | 7.31 | 8.18 | 7.48 | 89.76 | 61.22 | 66.40 | 119.00 | 84.09 |

### 12.5.4 Experiment 4: ECGA Crossover Operators

In Experiment 4, we explore the significance of the different operators involved in crossover; linear sum, monomial and OWA. Figure 6 reports the best fitness for the following scenarios; crossover with (i) all three operations, (ii) just linear sum, (iii) just monomial, (iv) just OWA, and (v) linear sum and OWA. As we can see, OWA, linear sum and OWA, and all three exhibit almost identical trends with best results for $e_1$, $e_2$, and $e_3$. However, OWA's performance appears to degrade considerably relative to the others for $e_4$. For relatively large problems, e.g., $63$ variables, the best score trends for linear sum and OWA are very close (if not identical) for all four functions. Based on the experimental findings, and guided by intuition, we are led to believe that individually none of these operators are universally best. The takeaway from this experiment is that the selection of appropri-

ate operators is important and adding more operators may have diminishing results with increasing computational complexity.




(a) $e_1$ for three inputs (b) $e_2$ for inputs (c) $e_3$ for three inputs

(d) $e_4$ for three inputs (e) $e_1$ for six inputs (f) $e_2$ for six inputs

(g) $e_3$ for six inputs (h) $e_4$ for six inputs

Figure 12.5
Experiment 4. Performance comparison of linear sum, OWA and monomial operators in ECGA crossover.

## 12.6 Conclusion

Herein, we proposed new operators for an efficient evolutionary algorithm. Specifically, these operators are focused on learning the Choquet integral. As such, the under-

lying problem that we want to solve is minimization of some function error relative to a large number of inequality constraints (due to the monotonicity and boundary conditions of the fuzzy measure). To this end, we introduced new theory and algorithms with a solid geometric interpretation. Our method more-or-less allows us to "bypass" our constraints and run an evolutionary algorithm in a smooth and uninterrupted fashion. The method is applicable to a number of problems, but where it excels is in cases of larger numbers of inputs and on complicated (non-convex) error functions. Experiments demonstrated superiority in accuracy and computational performance relative to prior work on different ChI and a benchmark non-ChI function (the Rastrigin function).

In future work, we will extend our theory (operators) beyond the scope of optimizing the ChI. Whereas we demonstrated its operation on the Rastrigin function, we will go further and study the more abstract problem of efficient solutions to complex functions with large numbers of equality and inequality constraints. Also, we showed in [111] that most data-driven problems do not involve all variables in the fuzzy measure. As such, we will follow our ideas in [111] and look for a way to "compress" and further improve the efficiency of our evolutionary optimization in a data set/variable specific fashion. Last, many problems involve non-real-valued (e.g., intervals, type-1 sets, type-2 sets, etc.) integrands and/or measures. In the future we will explore the extension of our operators for uncertain inputs and measures.

# CHAPTER XIII

# CONCLUSIONS

## 13.1 Contributions

Due to an increasing reliance on multiple sources of data/information in a wide variety of applications, the demand for aggregation is not going away. To date, a limited amount of work has focused on the efficiency of learning algorithms. In this dissertation I present several approaches to efficiently represent and optimize the ChI.

The learning of the ChI in its standard form becomes quickly intractable due to its exponential increase of variables with respect to input size. Therefore, most applications either resort to simpler and relatively restrictive integrals with fewer variables–such as Sugeno $\lambda$-measure or $k$-additive–or they use fixed operators like the max, min, majority voting, etc. In Chapter VIII, I proposed the identification and optimization of the variables supported by data. I also proposed an imputation strategy for data-unsupported variables, which can be fixed or learned. From the experiments conducted on synthetic data sets, it was demonstrated that the proposed method outperforms $k$-additive measure and can yield better results than the standard ChI when suitable imputation function is used.

In Chapter XII, I presented an efficient GA for optimizing ChI problems. As GAs do not naturally provide a mechanism to handle constraints directly, numerous indirect ways have been introduced. Since the ChI has $N(2^N - 1)$ constraints for $N$ inputs, its opti-

mization is costly with generalized constraint handling approaches. Several GAs, specifically for learning the ChI, have recently been proposed such as MICIGA and FVFMGA. MICIGA uses large-scale mutation to search the space while small-scale mutation for convergence. FVFMGA fixes the violated chromosome in a bottom-up fashion. Experiments result show that ECGA is computationally efficient in compared with MICIGA and FVFMGA for a large number of inputs.

In Chapter II, the BChI was introduced, where the variables in the BFM lie in $\{0, 1\}$ instead of $[0, 1]$. This representation offers savings in computation and storage by drastically reducing the search space. Moreover, the ChI computation requires only one variable. The BChI is a natural fit for some applications and can be approximated for others. An efficient learning algorithm for the BChI was presented in Chapter IX, which shows a huge saving in the number of variables required to learn, represent, and compute the BChI.

In Chapter V, I proposed a dimensionality reduction technique using a visual clustering algorithm, CLODD, which is inspired by the way human do clustering from an image. The proposed method can perform both contiguous and non-contiguous band grouping in supervised or unsupervised manner. The experiment results on the Indian Pines data set show that overall contiguous clustering has superior performance, which is intuitive as it carries more information than the non-contiguous one.

In Chapter VI, I studied the impact of varying Lp-norm for feature level fusion using MKL in hyperspectral image processing. While L1-norm MKL (or sparse MKL) is suitable for kernel selection from noisy features, higher norm MKL can significantly improve results via dense aggregation of diverse and complementary features. However, MKL works

in hyperspectral images heavily favored the use of L1-norm MKL and to the best of my knowledge that explored Lp-norm MKL. In the proposed method, two diverse feature sets using two different proximity measures, square of Euclidean and correlation, were used to extract features, which were then fused with two kernels, RBF and correlation, using MKL. Experiments conducted on the Indian Pines dataset demonstrate that in most cases, higher norm (e.g., $p$=100) yields higher classification accuracy than the L1-norm MKL.

## 13.2 Future Research Directions

This dissertation provides a framework for multiple directions of future work in the area of efficient learning and representation of aggregation operators. For example:

**Data-driven learning:** In EChIGA, the problem is partitioned into data supported and unsupported variables. However, for data supported not all variables are supported to the same degree. For example, some variables could have hundreds of samples whereas others may have a single sample. It would be interesting to study how to combat different degrees of "data supported" variables in learning. In addition, I plan to explore different error and/or penalty functions and associated optimization algorithms [111].

**Efficient GA:** In future work, I will extend the theory (operators) beyond the scope of optimizing the ChI. As shown in [111], most data-driven problems do not involve all variables in the fuzzy measure. As such, I will follow our ideas in [111] and look for a way to "compress" and further improve the efficiency of evolutionary optimization in a data set/variable specific fashion. Last, many problems involve non-real-valued (e.g., intervals,

type-1 sets, type-2 sets, etc.) integrands and/or measures. In the future I will explore the extension of our operators for uncertain inputs and measures [109].

**Aggregation under uncertainty:** This article is just a first step towards missing data and the FI. In future work, I will expand our scope to include both missing integrand ($h$) and missing FM ($\mu$), the latter only slightly touched on in our case study. I will also attempt to simultaneously solve for the FM in conjunction with how to normalize it (versus specify the normalization). I will also explore pathways involving "up sampling" (type increasing). In future work I will study interval and fuzzy set valued label driven learning, or learning of such data in light of scalar-valued inputs [110].

**Lp-norm MKL feature fustion:** In the future, I will investigate a search procedure for MKL parameter selection, including kernel type and associated parameters (a critical aspect of MKL that is typically overlooked due to complexity). Last, currently all features produced by band grouping are used. However, sometimes some bands (or band groups) are not useful for a task at hand and performance can be raised if the feature selection is performed before fusion [106].

**Fusion in deep learning:** In the future, my plan is to migrate the ChI solution into a strictly neural representation for optimization and speed. Furthermore, fusion neurons at various layers in the network will be explored. Future work will also include simultaneously learning the DCNNs and our fusion operators (they are learned independently herein) [13].

This dissertation proposed efficient methods to represent and learn aggregation operators that I believe would be useful for many applications. There remain many more challenges in numerous aspects of fusion such as aggregation of heterogeneous inputs, aggregation under uncertainty, and Choquistic neuron representation. I believe it would be well worth to study these problems.

REFERENCES


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[2] M. Al Boni, D. T. Anderson, and R. L. King, "Constraints preserving genetic algorithm for learning fuzzy measures with an application to ontology matching," *Advance Trends in Soft Computing*, Springer, 2014, pp. 93–103.

[3] M. Al Boni, D. T. Anderson, and R. L. King, "Hybrid Measure of Agreement and Expertise for Ontology Matching in Lieu of a Reference Ontology," *International Journal of Intelligent Systems*, vol. 31, no. 5, 2016, pp. 502–525.

[4] R. Alain, R. Francis, C. Phane, and S. Yves, "Simple MKL," *Journal of Machine Learning Research*, vol. 9, 2008, pp. 2491–2521.

[5] D. Anderson, G. Scott, M. Islam, B. Murray, and R. Marcum, "Fuzzy Choquet Integration of Deep Convolutional Neural Networks for Remote Sensing," *Computational Intelligence for Pattern Recognition*, Witold and Shyi-Ming, eds., Springer Berlin Heidelberg, 2018, pp. pp–pp.

[6] D. T. Anderson, P. Elmore, F. Petry, and T. C. Havens, "Fuzzy Choquet integration of homogeneous possibility and probability distributions," *Information Sciences*, vol. 363, 2016, pp. 24 – 39.

[7] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. F. Anderson, and D. J. Wescott, "Sugeno fuzzy integral generalizations for Sub-normal Fuzzy set-valued inputs," *2012 IEEE International Conference on Fuzzy Systems*, June 2012, pp. 1–8.

[8] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. F. Anderson, and D. J. Wescott, "Extension of the Fuzzy Integral for General Fuzzy Set-Valued Information," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 6, Dec 2014, pp. 1625–1639.

[9] D. T. Anderson, M. Islam, R. King, N. H. Younan, J. R. Fairley, S. Howington, F. Petry, P. Elmore, and A. Zare, "Binary fuzzy measures and Choquet integration for multi-source fusion," *Military Technologies (ICMT), 2017 International Conference on*. IEEE, 2017, pp. 676–681.

[10] D. T. Anderson, J. M. Keller, M. Anderson, and D. J. Wescott, "Linguistic description of adult skeletal age-at-death estimations from fuzzy integral acquired fuzzy sets," *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, June 2011, pp. 2274–2281.

[11] D. T. Anderson, J. M. Keller, and T. C. Havens, "Learning fuzzy-valued fuzzy measures for the fuzzy-valued sugeno fuzzy integral," *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2010, pp. 502–511.

[12] D. T. Anderson, S. R. Price, and T. C. Havens, "Regularization-based learning of the Choquet integral," *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2519–2526.

[13] D. T. Anderson, G. Scott, M. A. Islam, B. Murray, R. Marcum, and W. Starms, *Fuzzy Choquet Integration of Deep Convolutional Neural Networks for Remote Sensing*, Springer-Verlag, to appear.

[14] M. F. Anderson, D. T. Anderson, and D. J. Wescott, "Estimation of adult skeletal age-at-death using the Sugeno fuzzy integral," *American journal of physical anthropology*, vol. 142, no. 1, 2010, pp. 30–41.

[15] S. Angilella, S. Corrente, and S. Greco, "Stochastic multiobjective acceptability analysis for the Choquet integral preference model and the scale construction problem," *European Journal of Operational Research*, vol. 240, no. 1, 2015, pp. 172–182.

[16] S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo, "Assessing non-additive utility for multicriteria decision aid," *European Journal of Operational Research*, vol. 158, no. 3, 2004, pp. 734–744.

[17] S. Angilella, S. Greco, and B. Matarazzo, "Non-additive robust ordinal regression: A multiple criteria decision model based on the Choquet integral," *European Journal of Operational Research*, vol. 201, no. 1, 2010, pp. 277–288.

[18] S. Auephanwiriyakul, J. M. Keller, and P. D. Gader, "Generalized Choquet fuzzy integral fusion," *Information Fusion*, vol. 3, no. 1, 2002, pp. 69–85.

[19] D. Avis and K. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry*, vol. 8, no. 3, 1992, pp. 295–313.

[20] I. B. Aydilek and A. Arslan, "A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm," *Information Sciences*, vol. 233, 2013, pp. 25–35.

[21] J. E. Ball, D. T. Anderson, and C. S. Chan, "A Comprehensive Survey of Deep Learning in Remote Sensing: Theories, Tools and Challenges for the Community," *Journal of Applied Remote Sensing*, 2017.

[22] J. E. Ball, D. T. Anderson, and S. Samiappan, "Hyperspectral band selection based on the aggregation of proximity measures for automated target detection," *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, pp. 908814–908814.

[23] J. E. Ball and L. Bruce, "Level Set Hyperspectral Segmentation: Near-Optimal Speed Functions using Best Band Analysis and Scaled Spectral Angle Mapper," *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*. IEEE, 2006, pp. 2596–2600.

[24] M. Bedworth, "Probability moderation for multilevel information processing," *Personal Communication*, 1994.

[25] M. Bedworth and J. O'Brien, "The Omnibus model: a new model of data fusion?," *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 4, 2000, pp. 30–36.

[26] G. Beliakov, "Construction of aggregation functions from data using linear programming," *Fuzzy Sets and Systems*, vol. 160, no. 1, 2009, pp. 65–75.

[27] G. Beliakov, S. James, and G. Li, "Learning Choquet-integral-based metrics for semisupervised clustering," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 3, 2011, pp. 562–574.

[28] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*, 1st edition, Springer Publishing Company, Incorporated, 2008.

[29] C. Bentes, D. Velotto, and S. Lehner, "Target classification in oceanographic SAR images with deep neural networks: Architecture and initial results," *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*. IEEE, 2015, pp. 3703–3706.

[30] P. Benvenuti, R. Mesiar, and D. Vivona, "Monotone set functions-based integrals," *Handbook of measure theory*, vol. 2, 2002, pp. 1329–1379.

[31] J. Bezdek, R. Hathaway, J. Huband, and M. Jacalyn, "Visual assessment of clustering tendency for rectangular dissimilarity matrices," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 5, 2007, pp. 890–903.

[32] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[33] L. Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, 1991.

[34] J. R. Boyd, *A discourse on winning and losing*, 1987.

[35] L. Brown, "Deep learning with GPUs," .

[36] B. Buckles and F. Petry, "Genetic Algorithms IEEE Computer Society Press," 1992.

[37] H. Bustince, G. Beliakov, G. P. Dimuro, B. Bedregal, and R. Mesiar, "On the definition of penalty functions in data aggregation," *Fuzzy Sets and Systems*, 2016.

[38] C. Chen, B. Zhang, H. Su, W. Li, and L. Wang, "Land-use scene classification using multi-scale completed local binary patterns," *Signal, Image and Video Processing*, vol. 10, no. 4, 2016, pp. 745–752.

[39] J. Chen, C. Richard, and P. Honeine, "Nonlinear unmixing of hyperspectral images based on multi-kernel learning," *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2012 4th Workshop on*, June 2012, pp. 1–4.

[40] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[41] X. Chen, S. Xiang, C. L. Liu, and C. H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, 2014, pp. 1797–1801.

[42] J. Chiang, "Choquet fuzzy integral-based hierarchical networks for decision analysis," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 1, 1999, pp. 63–71.

[43] S. Cho and J. H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 2, 1995, pp. 380–384.

[44] G. Choquet, "Theory of capacities," *Annales de l'institut Fourier*. Institut Fourier, 1954, vol. 5, pp. 131–295.

[45] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[46] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, 2010, pp. 3207–3220.

[47] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering*, vol. 191, no. 11-12, 2002, pp. 1245–1287.

[48] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 160–167.

[49] C. Cortes, M. Mohri, and A. Rostamizadeh, "$\ell_2$ regularization for learning kernels," *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 109–116.

[50] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, 1995, pp. 273–297.

[51] D. Dai and W. Yang, "Satellite image classification via two-layer sparse coding with biased image representation," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 1, 2011, pp. 173–176.

[52] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proceedings of the IEEE*, vol. 85, no. 1, 1997, pp. 24–38.

[53] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, 2000, pp. 311–338.

[54] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, 1977, pp. 1–38.

[55] X. Du, A. Zare, J. M. Keller, and D. T. Anderson, "Multiple Instance Choquet integral for classifier fusion," *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 1054–1061.

[56] D. J. Dubois, *Fuzzy sets and systems: theory and applications*, vol. 144, Academic press, 1980.

[57] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, 2011, pp. 2121–2159.

[58] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, 1980, pp. 238–246.

[59] Q. Fu, X. Yu, X. Wei, and Z. Xue, "Semi-supervised classification of hyperspectral imagery based on stacked autoencoders," *Eighth International Conference on Digital Image Processing (ICDIP 2016)*. International Society for Optics and Photonics, 2016, pp. 100332B–100332B.

[60] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," *Competition and Cooperation in Neural Nets*, Springer, 1982, pp. 267–285.

[61] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, 1993, pp. 801–806.

[62] P. D. Gader, J. M. Keller, and B. N. Nelson, "Recognition technology for the detection of buried land mines," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 1, 2001, pp. 31–43.

[63] P. J. García-Laencina, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, 2010, pp. 263–282.

[64] J. Geng, J. Fan, H. Wang, X. Ma, B. Li, and F. Chen, "High-resolution SAR image classification via deep convolutional autoencoders," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, 2015, pp. 2351–2355.

[65] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," *Advances in neural information processing systems*, 1994, pp. 120–120.

[66] F. Glover, G. A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," *Management Science*, vol. 44, no. 3, 1998, pp. 336–345.

[67] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *The Journal of Machine Learning Research*, vol. 12, 2011, pp. 2211–2268.

[68] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[69] M. Grabisch, "Fuzzy integral in multicriteria decision making," *Fuzzy sets and Systems*, vol. 69, no. 3, 1995, pp. 279–298.

[70] M. Grabisch, "The application of fuzzy integrals in multicriteria decision making," *European journal of operational research*, vol. 89, no. 3, 1996, pp. 445–456.

[71] M. Grabisch, "K-order additive discrete fuzzy measures and their representation," *Fuzzy sets and systems*, vol. 92, no. 2, 1997, pp. 167–189.

[72] M. Grabisch, "Fuzzy integral for classification and feature extraction," *Fuzzy Measures and Integrals: Theory and Applications*, Springer-Verlag New York, Inc., 2000, pp. 415–434.

[73] M. Grabisch, *Fuzzy Measures and Integrals: Theory and Applications*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[74] M. Grabisch, "An axiomatization of the Shapley value and interaction index for games on lattices," *SCIS-ISIS*, 2004.

[75] M. Grabisch, I. Kojadinovic, and P. Meyer, "A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the Kappalab R package," *European journal of operational research*, vol. 186, no. 2, 2008, pp. 766–785.

[76] M. Grabisch, H. T. Nguyen, and E. A. Walker, *Fundamentals of uncertainty calculi with applications to fuzzy inference*, vol. 30, Springer Science & Business Media, 2013.

[77] M. Grabisch and J. Nicolas, "Classification by fuzzy integral: performance and tests," *Fuzzy sets and systems*, vol. 65, no. 2-3, 1994, pp. 255–271.

[78] M. Grabisch and M. Roubens, "An axiomatic approach to the concept of interaction among players in cooperative games," *International Journal of Game Theory*, vol. 28, no. 4, 1999, pp. 547–565.

[79] M. Grabisch and M. Roubens, "Application of the Choquet integral in multicriteria decision making," *Fuzzy Measures and Integrals-Theory and Applications*, 2000, pp. 348–374.

[80] M. Grabisch and M. Sugeno, "Multi-attribute classification using fuzzy integral," *IEEE International Conference on Fuzzy Systems*. IEEE, 1992, pp. 47–54.

[81] Y. Gu, G. Gao, D. Zuo, and D. You, "Model Selection and Classification With Multiple Kernel Learning for Hyperspectral Images via Sparsity," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 7, no. 6, 2014, pp. 2119–2130.

[82] Y. Gu, C. Wang, D. You, Y. Zhang, S. Wang, and Y. Zhang, "Representative multiple kernel learning for classification in hyperspectral imagery," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 50, no. 7, 2012, pp. 2852–2865.

[83] Y. Gu, Q. Wang, H. Wang, D. You, and Y. Zhang, "Multiple Kernel Learning via Low-Rank Nonnegative Matrix Factorization for Classification of Hyperspectral Imagery," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 8, no. 6, 2015, pp. 2739–2751.

[84] P. Gurram and H. Kwon, "Optimal sparse kernel learning in the empirical kernel feature space for hyperspectral classification," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 7, no. 4, 2014, pp. 1217–1226.

[85] T. C. Havens, D. T. Anderson, and J. M. Keller, "A fuzzy Choquet integral with an interval type-2 fuzzy number-valued integrand," *International Conference on Fuzzy Systems*, July 2010, pp. 1–8.

[86] T. C. Havens, D. T. Anderson, and C. Wagner, "Data-Informed Fuzzy Measures for Fuzzy Integration of Intervals and Fuzzy Numbers," *IEEE Trans. Fuzzy Syst.*, vol. PP, no. 99, 2014, pp. 1–1.

[87] T. C. Havens, D. T. Anderson, C. Wagner, H. Deilamsalehy, and D. Wonnacott, "Fuzzy integrals of crowd-sourced intervals using a measure of generalized accord," *IEEE International Conference on Fuzzy Systems*. IEEE, 2013, pp. 1–8.

[88] T. C. Havens and J. Bezdek, "An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, 2012, pp. 813–822.

[89] T. C. Havens, J. Bezdek, J. M. Keller, and M. Popescu, "Clustering in ordered dissimilarity data," *International Journal of Intelligent Systems*, vol. 24, no. 5, 2009, pp. 504–528.

[90] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[91] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, 2009, p. 5947.

[92] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, 2006, pp. 504–507.

[93] J. H. Holland, "Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence," *Ann Arbor, MI: University of Michigan Press*, 1975, pp. 439–444.

[94] J. H. Holland, "Adaptation in natural and artificial systems. 1975," *Ann Arbor, MI: University of Michigan Press and*, 1992.

[95] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, 1994, pp. 242–253.

[96] P. Honeine and C. Richard, "The angular kernel in machine learning for hyperspectral data classification," *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2010 2nd Workshop on*, June 2010, pp. 1–4.

[97] L. Hu, D. T. Anderson, and T. C. Havens, "Multiple kernel aggregation using fuzzy integrals," *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2013, pp. 1–7.

[98] L. Hu, D. T. Anderson, T. C. Havens, and J. M. Keller, *Efficient and Scalable Nonlinear Multiple Kernel Aggregation Using the Choquet Integral*, Springer International Publishing, 2014, pp. 206–215.

[99] W. Huang, L. Xiao, Z. Wei, H. Liu, and S. Tang, "A new pan-sharpening method with deep neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 5, 2015, pp. 1037–1041.

[100] J. P. Ignizio, *Goal programming and extensions*, vol. 26, Lexington Books Lexington, MA, 1976.

[101] J. P. Ignizio and C. Romero, "Goal programming," *Encyclopedia of information systems*, vol. 2, 2003, pp. 489–500.

[102] M. Imani and H. Ghassemian, "Band clustering-based feature extraction for classification of hyperspectral images using limited training samples," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 8, 2014, pp. 1325–1329.

[103] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine Learning*, 2015, pp. 448–456.

[104] M. G. Iskander, "A Suggested Approach for Solving Weighted Goal Programming Problem," *American Journal of Computational and Applied Mathematics*, 2012.

[105] M. Islam, D. T. Anderson, J. E. Ball, and N. H. Younan, "CLODD based band group selection," *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*. IEEE, 2016, pp. 36–39.

[106] M. Islam, D. T. Anderson, J. E. Ball, and N. H. Younan, "Fusion of diverse features and kernels using $l_p$-norm based multiple kernel learning in hyperspectral image processing," *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, August 2016.

[107] M. Islam, D. T. Anderson, X. Du, and W. C. Havens, Timothy C, "Efficient Binary Fuzzy Measure Representation and Choquet Integral Learning," *2018 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, to be submitted.

[108] M. Islam, D. T. Anderson, and T. C. Havens, "Multi-criteria based learning of the Choquet integral using Goal programming," *Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American*. IEEE, 2015, pp. 1–6.

[109] M. Islam, D. T. Anderson, F. Petry, and P. Elmore, "An Efficient Evolutionary Algorithm for Optimization of the Choquet Integral," *International Journal of Intelligent Systems*, under review.

[110] M. Islam, D. T. Anderson, F. Petry, D. Smith, and P. Elmore, "The fuzzy integral for missing data," *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–8.

[111] M. Islam, D. T. Anderson, A. J. Pinar, and T. C. Havens, "Data-Driven Compression and Efficient Learning of the Choquet Integral," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, 2017, pp. 1–1.

[112] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *Proc. of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[113] H. Jiang and W.-K. Ching, "Correlation Kernels for support vector machines classification with applications in cancer data," *Computational and mathematical methods in medicine*, vol. 2012, 2012.

[114] X. Jin and C. H. Davis, "Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks," *Image and Vision Computing*, vol. 25, no. 9, 2007, pp. 1422–1431.

[115] X. Jin and R. G. Reynolds, "Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. IEEE, 1999, vol. 3.

[116] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 579–584.

[117] J. M. Keller, P. Gader, and A. K. Hocaoglu, "Fuzzy integral in image processing and recognition," *Fuzzy Measures and Integrals: Theory and Applications*, Springer-Verlag New York, Inc., 2000, pp. 435–466.

[118] J. M. Keller, P. Gader, H. Tahani, J. Chiang, and M. Mohamed, "Advances in fuzzy integration for pattern recognition," *Fuzzy sets and systems*, vol. 65, no. 2-3, 1994, pp. 273–283.

[119] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , no. 6, 1985, pp. 693–699.

[120] J. M. Keller and J. Osborn, "A reward/punishment scheme to learn fuzzy densities for the fuzzy integral," *International Fuzzy Systems Association World Congress*, 1995, pp. 97–100.

[121] J. M. Keller and J. Osborn, "Training the fuzzy integral," *International Journal of Approximate Reasoning*, vol. 15, no. 1, 1996, pp. 1–24.

[122] 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, 2013, pp. 28–44.

[123] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *3rd International Conference for Learning Representations*, 2015.

[124] M. Kloft, U. Brefeld, P. Laskov, and S. Sonnenburg, "Non-sparse multiple kernel learning," *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[125] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm multiple kernel learning," *The Journal of Machine Learning Research*, vol. 12, 2011, pp. 953–997.

[126] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[127] C. Labreuche, "Construction of a Choquet integral and the value functions without any commensurateness assumption in multi-criteria decision making," *EUSFLAT Conf.*, 2011, pp. 90–97.

[128] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[129] M. Lee, D. Anderson, J. E. Ball, and J. White, *Background adaptive division filtering for hand-held ground penetrating radar*, SPIE, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI, 2016.

[130] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*, John Wiley & Sons, 2014.

[131] T. Magoc and F. Modave, "Optimization of the Choquet integral using genetic algorithm," *Constraint Programming and Decision Making*, Springer, 2014, pp. 97–109.

[132] M. Markin, C. Harris, M. Bernhardt, J. Austin, M. Bedworth, P. Greenway, R. Johnston, A. Little, and D. Lowe, "Technology foresight on data fusion and data processing," *The Royal Aeronautical Society*, 1997.

[133] A. Martínez-Usó, F. Pla, J. Sotoca, and P. García-Sevilla, "Clustering-based hyperspectral band selection using information measures," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, 2007, pp. 4158–4171.

[134] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, 1943, pp. 115–133.

[135] P. Melin, O. Mendoza, and O. Castillo, "Face recognition with an improved interval type-2 fuzzy logic sugeno integral and modular neural networks," *IEEE Transactions on systems, man, and cybernetics-Part A: systems and humans*, vol. 41, no. 5, 2011, pp. 1001–1012.

[136] A. Mendez-Vazquez and P. Gader, "Sparsity promotion models for the choquet integral," *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. IEEE, 2007, pp. 454–459.

[137] A. Mendez-Vazquez, P. Gader, J. M. Keller, and K. Chamberlin, "Minimum Classification Error Training for Choquet Integrals With Applications to Landmine Detection," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 1, Feb 2008, pp. 225–238.

[138] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods.," *Evolutionary Programming*, vol. 4, 1995, pp. 135–155.

[139] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model.," *Interspeech*, 2010, vol. 2, p. 3.

[140] P. Miranda, M. Grabisch, and P. Gil, "Axiomatic structure of¡ i¿ k¡/i¿-additive capacities," *Mathematical Social Sciences*, vol. 49, no. 2, 2005, pp. 153–178.

[141] K. Mohan, G. Van den Brock, A. Choi, and J. Pearl, "An Efficient Method for Bayesian Network Parameter Learning from Incomplete Data," *Causal Modeling and Machine Learning Workshop*, 2014, vol. 951, p. 2014.

[142] T. Murofushi and S. Soneda, "Techniques for reading fuzzy measures (III): interaction index," *9th Fuzzy System Symposium*. Sapporo,, Japan, 1993, pp. 693–696.

[143] T. Murofushi and M. Sugeno, "An interpretation of fuzzy measures and the Choquet integral as an integral with respect to a fuzzy measure," *Fuzzy sets and Systems*, vol. 29, no. 2, 1989, pp. 201–227.

[144] B. Murray, M. Islam, A. J. Pinar, T. C. Havens, D. T. Anderson, and G. Scott, "Explainable AI for Understanding Decisions and Data-Driven Optimization of the Choquet Integral," *Fuzzy Systems (FUZZ-IEEE), 2018 IEEE International Conference on*. IEEE, 2018.

[145] S. D. Newsam, "UC Merced Land Use Dataset,", http://vision.ucmerced.edu/datasets/landuse.html, 2010.

[146] H. Nguyen, V. Kreinovich, J. Lorkowski, and S. Abu, "Why Sugeno -Measures," *Technical Report: UTEP-CS-15-17*, 2015.

[147] C. Olsson, A. P. Eriksson, and F. Kahl, "Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming," *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[148] S. K. Pal and S. Mitra, *Neuro-fuzzy pattern recognition: methods in soft computing*, John Wiley & Sons, Inc., 1999.

[149] A. J. Pinar, D. T. Anderson, T. C. Havens, A. Zare, and T. Adeyeba, "Measures of the Shapley index for learning lower complexity fuzzy integrals," *Granular Computing*, 2017, pp. 1–17.

[150] A. J. Pinar, T. C. Havens, D. T. Anderson, and L. Hu, "Feature and decision level fusion using multiple kernel learning and fuzzy integrals," *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Aug 2015, pp. 1–7.

[151] A. J. Pinar, T. C. Havens, M. Islam, and D. T. Anderson, "Visualization and learning of the Choquet integral with limited training data," *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[152] A. J. Pinar, J. Rice, L. Hu, D. T. Anderson, and T. C. Havens, "Efficient Multiple Kernel Classification using Feature and Decision Level Fusion," *IEEE Trans. Fuzzy Syst.*, vol. PP, no. 99, 2016, pp. 1–1.

[153] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, 1964, pp. 1–17.

[154] S. R. Price, D. T. Anderson, C. Wagner, T. C. Havens, and J. M. Keller, "Indices for Introspection on the Choquet Integral," *Advance Trends in Soft Computing*, Springer, 2014, pp. 261–271.

[155] S. R. Price, B. Murray, L. Hu, D. T. Anderson, T. C. Havens, R. H. Luke, and J. M. Keller, "Multiple kernel based feature and decision level fusion of iECO individuals for explosive hazard detection in FLIR imagery," *SPIE*, 2016, vol. 9823, pp. 98231G–98231G–11.

[156] J. R. Quinlan, *C4. 5: programs for machine learning*, Elsevier, 2014.

[157] S. Rajurkar and N. K. Verma, "Developing deep fuzzy network with Takagi Sugeno fuzzy inference system," *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2017, pp. 1–6.

[158] M. Ramoni and P. Sebastiani, "Learning Bayesian networks from incomplete databases," *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1997, pp. 401–408.

[159] M. Ramoni and P. Sebastiani, "Robust learning with missing data," *Machine Learning*, vol. 45, no. 2, 2001, pp. 147–170.

[160] D. Rumsfeld, "Defense department briefing," Feburary 2002.

[161] M. Saar-Tsechansky and F. Provost, "Handling missing values when applying classification models," *Journal of machine learning research*, vol. 8, no. Jul, 2007, pp. 1623–1657.

[162] G. J. Scott and D. T. Anderson, "Importance-weighted multi-scale texture and shape descriptor for object recognition in satellite imagery," *2012 IEEE International Geoscience and Remote Sensing Symposium*, July 2012, pp. 79–82.

[163] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis, "Training Deep Convolutional Neural Networks for Land-Cover Classification of High-Resolution Imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 4, 2017, pp. 549–553.

[164] G. J. Scott, R. A. Marcum, C. H. Davis, and T. W. Nivin, "Fusion of Deep Convolutional Neural Networks for Land Cover Classification of High-Resolution Imagery," *IEEE Geoscience & Remote Sens. Letters*, 2017.

[165] A. N. Shulsky and G. J. Schmitt, *Silent warfare: understanding the world of intelligence*, Potomac Books, Inc., 2002.

[166] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[167] R. E. Smith, D. T. Anderson, A. Zare, J. E. Ball, B. Alvey, J. R. Fairley, and S. E. Howington, "Genetic Programming Based Choquet Integral for Multi-Source Fusion," *IEEE Int. Conf. Fuzzy Systems*, Jul. 2017.

[168] R. E. Smith, D. T. Anerson, J. E. Ball, A. Zare, and B. Alvey, "Aggregation of Choquet integrals in GPR and EMI for handheld platform-based explosive hazard detection," *Proc. SPIE 10182, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*, 2017.

[169] A. J. Smola, S. Vishwanathan, and T. Hofmann, "Kernel Methods for Missing Variables.," *AISTATS*. Citeseer, 2005.

[170] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 129–136.

[171] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, and V. Franc, "The SHOGUN machine learning toolbox," *The Journal of Machine Learning Research*, vol. 11, 2010, pp. 1799–1802.

[172] E. Sperner, "Ein satz über untermengen einer endlichen menge," *Mathematische Zeitschrift*, vol. 27, no. 1, 1928, pp. 544–548.

[173] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, 2014, pp. 1929–1958.

[174] A. N. Steinberg, C. L. Bowman, and F. E. White, "Revisions to the JDL data fusion model," *Handbook of Data Fusion*, 1999.

[175] M. Sugeno, "Fuzzy measure and fuzzy integral," *Fuzzy Measure and Fuzzy Integral*, vol. 8, no. 2, 1972, pp. 94–102.

[176] M. Sugeno, *Theory of fuzzy integrals and its applications*, doctoral dissertation, Tokyo Institute of Technology, 1974.

[177] C. Sung-Bae, "Fuzzy Aggregation of Modular Neural Networks With Ordered Weighted Averaging Operators," *International Journal of Approximate Reasoning*, vol. 13, no. 4, 1995, pp. 359–375.

[178] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *International Conference on Machine Learning*, 2013, pp. 1139–1147.

[179] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[180] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[181] H. Tahani and J. M. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 20, no. 3, 1990, pp. 733–741.

[182] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, , no. 1, 1985, pp. 116–132.

[183] A. F. Tehrani, W. Cheng, K. Dembczyński, and E. Hüllermeier, "Learning monotone nonlinear models using the Choquet integral," *Machine Learning*, vol. 89, no. 1-2, 2012, pp. 183–211.

[184] A. F. Tehrani, W. Cheng, and E. Hüllermeier, "Choquistic Regression: Generalizing Logistic Regression using the Choquet Integral.," *EUSFLAT Conf.*, 2011, pp. 868–875.

[185] A. F. Tehrani, W. Cheng, and E. Hullermeier, "Preference learning using the choquet integral: The case of multipartite ranking," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, 2012, pp. 1102–1113.

[186] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *Coursera: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012, pp. 26–31.

[187] L. Tomlin, D. T. Anderson, C. Wagner, T. C. Havens, and J. M. Keller, *Fuzzy Integral for Rule Aggregation in Fuzzy Inference Systems*, Springer International Publishing, 2016, pp. 78–90.

[188] W. T. Trotter, *Combinatorics and partially ordered sets: Dimension theory*, vol. 6, JHU Press, 2001.

[189] D. Tuia, G. Camps-Valls, G. Matasci, and M. Kanevski, "Learning relevant image features with multiple-kernel classification," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 10, 2010, pp. 3780–3791.

[190] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 2015, pp. 689–692.

[191] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proceedings of the 25th International Conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[192] C. Wagner, D. Anderson, and T. Havens, "Generalization of the Fuzzy Integral for discontinuous interval- and non-convex interval fuzzy set-valued inputs," *IEEE International Conference on Fuzzy Systems*, July 2013, pp. 1–8.

[193] C. Wagner and D. T. Anderson, "Extracting meta-measures from data for fuzzy aggregation of crowd sourced information," *IEEE International Conference on Fuzzy Systems*. IEEE, 2012, pp. 1–8.

[194] C. Wagner, S. Miller, J. M. Garibaldi, D. T. Anderson, and T. C. Havens, "From Interval-Valued Data to General Type-2 Fuzzy Sets," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 2, April 2015, pp. 248–269.

[195] X. Wang, A. Chen, and H. Feng, "Upper integral network with extreme learning mechanism," *Neurocomputing*, vol. 74, no. 16, 2011, pp. 2520–2525.

[196] J. White, D. T. Anderson, J. E. Ball, and B. Parker, *Curvelet filter based prescreener for explosive hazard detection in hand-held ground penetrating radar*, SPIE, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI, 2016.

[197] F. E. White Jr, "Data fusion lexicon, joint directors of laboratories," *Technical panel for C*, vol. 3, 1987.

[198] Y. Won, P. D. Gader, and P. C. Coffield, "Morphological shared-weight networks with applications to automatic target recognition," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, 1997, pp. 1195–1203.

[199] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," *Advances in Neural Information Processing Systems*, 2014, pp. 1790–1798.

[200] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," *Proc. Int. Conf. Machine Learning*, 2010, pp. 1175–1182.

[201] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 18, no. 1, 1988, pp. 183–190.

[202] R. R. Yager, "Applications and extensions of OWA aggregations," *International Journal of Man-Machine Studies*, vol. 37, no. 1, 1992, pp. 103–122.

[203] R. Yang, Z. Wang, P. Heng, and K. Leung, "Classification of heterogeneous fuzzy data by Choquet integral with fuzzy-valued integrand," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 5, 2007, pp. 931–942.

[204] Y. Yang and S. Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, 2010, p. 666.

[205] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and computational Applications*, vol. 10, no. 1, 2005, pp. 45–56.

[206] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[207] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sensing Letters*, vol. 6, no. 6, 2015, pp. 468–477.

[208] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, 1965, pp. 338–353.

[209] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[210] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

[211] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2018–2025.

[212] J. Zhai, H. Xu, and Y. Li, "Fusion of extreme learning machine with fuzzy integral," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 21, no. 2, 2013, pp. 23–34.

[213] Y. Zhang, H. L. Yang, S. Prasad, E. Pasolli, J. Jung, and M. Crawford, "Ensemble multiple kernel active learning for classification of multisource remote sensing data," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 8, no. 2, 2015, pp. 845–858.

[214] C. Zhong, W. Pedrycz, D. Wang, L. Li, and Z. Li, "Granular data imputation: A framework of Granular Computing," *Applied Soft Computing*, vol. 46, 2016, pp. 307–316.