Michigan Technological University

Michigan Technological University
Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2021

EXPLAINABLE FEATURE- AND DECISION-LEVEL FUSION

Siva Krishna Kakula
Michigan Technological University, skakula@mtu.edu

Copyright 2021 Siva Krishna Kakula

Recommended Citation

Kakula, Siva Krishna, "EXPLAINABLE FEATURE- AND DECISION-LEVEL FUSION", Open Access Dissertation, Michigan Technological University, 2021.
https://doi.org/10.37099/mtu.dc.etdr/1185

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

 Part of the Data Science Commons, and the Theory and Algorithms Commons

EXPLAINABLE FEATURE- AND DECISION-LEVEL FUSION

By

Kakula Siva Krishna

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

© 2021 Kakula Siva Krishna

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Science.

Department of Computer Science

Dissertation Advisor:	*Dr. Timothy Havens*
Committee Member:	*Dr. Keith Vertanen*
Committee Member:	*Dr. Laura Brown*
Committee Member:	*Dr. Anthony Pinar*
Department Chair:	*Dr. Linda Ott*

Contents

List of Figures

List of Tables

List of Abbreviations

ChI	Choquet Fuzzy Integral
FM	Fuzzy Measure
CIR	Choquet Integral Regression
FI	Fuzzy Integral
DeFIMKL	Decision-level Fuzzy Integral Multiple Kernel Learning
OWA	Ordered Weighted Average
LOS	Linear Order Statistic
ELOS	Extended Linear Order Statistic
MKL	Multiple Kernel Learning
SVM	Support Vector Machine
BC	Bounded Capacity
QP	Quadratic Programming
SSE	Sum of Squared Error
MSE	Mean Squared Error

Preface

Some chapters of this dissertation contain published material. The following list indicates which publications were used along with notes on author contributions.

Chapter 2

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (July 2020). "Extended Linear Order Statistic," IEEE International Conference on Fuzzy Systems.

S.K. Kakula is the leading researcher in this work and is the corresponding author. The research was performed under the guidance of T.C. Havens and the ideas proposed in the paper stemmed from conversations among all authors.

Chapter 3

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (July 2020). "Choquet Integral Ridge Regression," IEEE International Conference on Fuzzy Systems.

The ideas presented in this paper are the result of discussions between all listed authors. S.K. Kakula is the corresponding author for this paper and generated the experimental results. A.J. Pinar and T.C. Havens contributed to the theoretical background.

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (Dec 2020). "Visualization and Analysis Tools for Explainable Choquet Integral Regression," IEEE Symposium Series on Computational Intelligence.

S.K. Kakula is the leading researcher in this work and is the corresponding author. The research was performed under the guidance of T.C. Havens, and the ideas proposed in the paper stemmed from conversations among all authors.

Chapter 4

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (Sep 2020). "Online Learning of the Fuzzy Choquet Integral," IEEE International Conference on Systems, Man, and Cybernetics (SMC).

S.K. Kakula is the leading researcher in this work and is the corresponding author. The research was performed under the guidance of T.C. Havens and the ideas proposed in the paper stemmed from conversations among all listed authors.

Chapter 5

S.K. Kakula, A.J. Pinar, Muhammed A. Islam, D.T. Anderson, and T.C. Havens (2020). "Novel Regularization for Learning the Fuzzy Choquet Integral with Limited Training Data," IEEE Transactions on Fuzzy Systems.

The ideas presented in this paper are the result of discussions between all listed authors. S.K. Kakula is the corresponding author for this paper and generated the experimental results. A.J. Pinar and T.C. Havens contributed to the theoretical background.

S.K. Kakula, T.C. Havens, A.J. Pinar, Muhammed A. Islam, D.T. Anderson, Andrew Buck, and Tim Wilkin. "Learning the Fuzzy Choquet Integral in the Presence of Uncertainty Caused by Limited Training Data Volume and Variety" Under Review - IEEE International Conference on Fuzzy Systems.

S.K. Kakula is the leading researcher in this work and is the corresponding author. The research was performed under the guidance of T.C. Havens and the ideas proposed in the paper stemmed from conversations among all listed authors.

Chapter 6

S.K. Kakula, T.C. Havens, and D.T. Anderson. "Online Sequential Learning of Monotonic Fuzzy Measures in the Choquet Integral," under review, IEEE International Conference on Fuzzy Systems.

The ideas presented in this paper are the result of discussions between all listed authors. S.K. Kakula is the corresponding author for this paper and generated the experimental results. T.C. Havens and D.T. Anderson contributed to the theoretical background.

Abstract

Information fusion is the process of aggregating knowledge from multiple data sources to produce more consistent, accurate, and useful information than any one individual source can provide. In general, there are three primary sources of data/information: humans, algorithms, and sensors. Typically, objective data—e.g., measurements—arise from sensors. Using these data sources, applications 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. There is a great need to discover new theories to combine and analyze heterogeneous data arising from one or more sources.

The work collected in this dissertation addresses the problem of feature- and decision-level fusion. Specifically, this work focuses on *fuzzy choquet integral* (ChI)-based data fusion methods. Most mathematical approaches for data fusion have focused on combining inputs relative to the assumption of independence between them. However, often there are rich interactions (e.g., correlations) between inputs that should be exploited. The ChI is a powerful aggregation tool that is capable modeling these interactions. Consider the fusion of m sources, where there are 2^m unique subsets (interactions); the ChI is capable of learning the worth of each of these possible source subsets. However, the complexity of

fuzzy integral-based methods grows quickly, as the number of trainable parameters for the fusion of m sources scales as 2^m. Hence, we require a large amount of training data to avoid the problem of over-fitting. This work addresses the over-fitting problem of ChI-based data fusion with novel regularization strategies. These regularization strategies alleviate the issue of over-fitting while training with limited data and also enable the user to consciously push the learned methods to take a predefined, or perhaps known, structure. Also, the existing methods for training the ChI for decision- and feature-level data fusion involve *quadratic programming* (QP). The QP-based learning approach for learning ChI-based data fusion solutions has a high space complexity. This has limited the practical application of ChI-based data fusion methods to six or fewer input sources. To address the space complexity issue, this work introduces an online training algorithm for learning ChI. The online method is an iterative gradient descent approach that processes one observation at a time, enabling the applicability of ChI-based data fusion on higher dimensional data sets.

In many real-world data fusion applications, it is imperative to have an explanation or interpretation. This may include providing information on what was learned, what is the worth of individual sources, why a decision was reached, what evidence process(es) were used, and what confidence does the system have on its decision. However, most existing machine learning solutions for data fusion are "black boxes," e.g., deep learning. In this work, we designed methods and metrics that help with answering these questions of interpretation, and we also developed visualization methods that help users better understand the machine learning solution and its behavior for different instances of data.

Chapter 1

Introduction

Information fusion is the process of aggregating knowledge from multiple data sources to produce more consistent, accurate, and useful information than any of the individual sources can provide. Humans are a great example of this process—we constantly fuse the sensory information from sources such as vision, hearing, taste, smell, and touch to perform the day to day tasks. Many important civilian and defense applications including weather forecasting, transportation management, battlefield assessment, target identification and classification are enhanced with the use of sensor and data fusion. One of the most well-known data fusion classification systems provided by Dasarathy [32] comprises five categories: i) raw data-level fusion: *data in-data out* (DAI-DAO), fusion of raw data from the sources to extract higher-level features; ii) *data in-feature out* (DAI-FEO), fusion of features to extract more consistent and informative features; iii) *feature in-feature*

out (FEI-FEO), fusion of features to produce an output decision; iv) *feature in-decision out* (FEI-DEO), fusion of decisions to obtain a new more useful and informed decision; and v) *decision in-decision* out (DEI-DEO). In this work, we developed novel data fusion methods for last two categories (FEI-DEO and DEI-DEO). This includes the development of novel algorithms for feature and decision-level fusion, evaluation of the algorithms using real-world data sets, and comparison of performance with the existing state-of-the-art algorithms. In addition to the improved performance, we also focus on trainability of algorithms with a limited amount of training data, and the explainability of the learned model as well as the output decisions.

1.1 Motivation

Typically, objective data—e.g., measurements—arise from sensors. These data may or may not be associated with uncertainty (inherent or in terms of what the sensor records). On the other hand, information, which could be objective or subjective, from humans is often riddled with uncertainty. Common culprits include randomness, measurement error, ambiguity, and vagueness, to name a few. In addition to this uncertainty, great complexity and diversity exists in both single and multi-source environments. Heterogeneity manifests itself in numerous ways, e.g., type and properties of uncertainty present and granularity of the data/information provided. Using these data sources, applications 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. There is a great need to discover new theories to combine and analyze heterogeneous data arising from one or more sources.

1.1.1 Explainability

To date, numerous ways have been created to learn a fusion solution from data. While this has resulted in significant leaps in numerous applications, explainability has not witnessed a similar growth. Most machine learning solutions for data fusion are "black boxes," e.g., deep learning. In many real-world applications, it is imperative to have an explanation; this may include providing information such as what was learned, what is the worth of individual sources, why a decision was reached, what evidence process(es) were used, and what confidence does the system have about its decision. This work introduces methods and metrics that help with answering these questions, and visualization methods that help the user better understand the machine learning solution and its behavior for different instances of data.

1.1.2 Learning with limited training data

One problem with learning an aggregation solution from data is that it often results in solutions that are overly complex and computationally expensive to implement. It also runs the risk of over-fitting and the quality of that solution depends on the size and diversity of the training data. The problem of over-fitting is more pronounced when the amount of training data is limited. While we want to achieve the highest performance possible, we also desire a simple solution—one that is more generalized to perform equally well on unseen data. We also desire a solution that requires the fewest computational resources (e.g., memory and processing), has the smallest form factor and energy consumption, etc. In this work, for the newly proposed methods of data fusion, several novel regularization strategies were introduced to alleviate the issue of over-fitting while training with limited data.

1.1.3 Leveraging rich interactions between sources

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 m inputs, there are 2^m possible subsets of source combinations to consider.

This work, with effective regularization strategies and visualization methods, enhances the fuzzy Choquet integral-based data fusion methods that enable us to leverage these interactions between the sources.

1.2 Background

While there are many categories of data fusion, this work focuses on two classes—decision-level fusion and feature-level fusion. Decision-level fusion methods aggregate the outputs from multiple decision makers to produce an overall fused decision, whereas feature-level fusion methods fuse multiple input features to produce a final output. Both these categories of data fusion have found their application in a wide range of applications. This work primarily focuses on *fuzzy integral* (FI)-based aggregation methods. Specifically, the *choquet fuzzy integral* (ChI)-based decision- and feature-level fusion methods. The aggregation using ChI is parametrized using a *fuzzy measure* (FM). This section provides the relevant background.

1.2.1 Fuzzy Integrals and Fuzzy Measures

FMs and FIs are used in several applications, e.g., classification [34, 35, 47, 55, 57, 80, 81, 82, 83, 86], pattern recognition [14, 41, 73], *multicriteria decision making* (MCDM)

[8, 9, 20, 25, 45, 76, 79, 117], forensic science [3], regression [59, 114, 115], and other applications [34, 66, 75]. One of the key challenges with using an FM is the assignment of values for its variables. While we might manage to manually specify the FM values for a small set of sources, since the number of trainable FM parameters scale as 2^m, where m is the number of input sources, it is virtually impossible to manually specify the FM values for a large collection of sources. To address this problem, several automatic methods have been proposed. The Sugeno λ-measure [49] and the S-decomposable measure [37] build the measure from the densities[1]. Methods that build the measure by using the training data include genetic algorithms [7], Gibbs sampling [85], *quadratic programming* (QP) [54], gradient descent [72], penalty/reward [71], and linear programming [12]. Other works [60, 61, 124] 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.

1.2.2 Fuzzy measures

Consider a measurable space as the tuple (X, Ω), where X is a set (typically of information sources or evidence [49]) and Ω is a σ-algebra, such that

P1. $X \in \Omega$;

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

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

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

An FM is a set-valued function, $g : \Omega \to [0, 1]$, with the following properties[112]:

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

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

There is an additional property that guarantees continuity for the case where Ω is an infinite set, however, in practice, Ω is finite and thus this property is unnecessary. FMs provide us with a convenient way to quantify the worth of combinations of sources, and FIs can be applied over FMs to aggregate the information from these sources.

1.2.3 Fuzzy integrals

There are many forms of the FI; see [34, 49, 51] for a detailed discussion. Several previous works [7, 10, 48, 70] have explored FIs as a tool for evidence fusion, and several generalizations of FIs are proposed in [13, 20, 35, 78, 80, 81, 83, 87]. The FM provides the expected worth of each subset of the sources, and the FIs use this to combine information from the sources while accounting for both the support of the evidence as well as the expected worth. In this dissertation, we focus on the Choquet fuzzy integral proposed by

Murofushi and Sugeno [27, 91]. Let $h : X \to \mathcal{R}$ be a real-valued function that represents the evidence or support of a particular hypothesis.[2] The discrete (finite Ω) Choquet FI is defined as

$$\int_C h \circ g = C_g(h) = \sum_{i=1}^{m} h(x_{\pi(i)}) \left[g(A_i) - g(A_{i-1}) \right], \tag{1.1}$$

where m is the number of input sources, and π 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 $g(A_0) = 0$ [54, 111]. Detailed treatments of the properties of FIs can be found in [45, 54, 111].

1.2.4 Visualizing the FI

This section presents the visualization approach for FMs introduced in [100]. A convenient method to visualize an FM is to represent it as a lattice (i.e., Hasse diagram); Figure 1.1 shows the lattice of an FM for the case of three sources ($n = 3$). The size of individual nodes (FM variables) in the FM lattice are scaled proportional to their FM values, and the lattice demonstrates the monotonicity constraints since the nodes at higher levels are larger—or at least as large—than those below.

The FM lattice is a great tool for visualization of the FM and it provides us with an intuition on the worth of different combinations of sources. However, the lattice alone does not give us the insight into how the Choquet integral at (1.1) utilizes the lattice to combine the

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

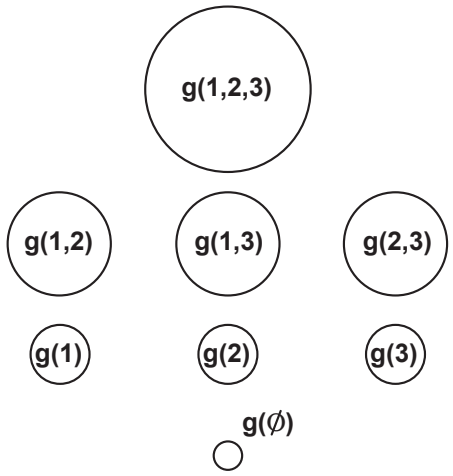

Figure 1.1: Lattice of FM elements for $m = 3$. Monotonicity (P5) is illustrated by the size of each node, i.e., $g(\{x_1\}) \leq g(\{x_1, x_2\})$ as $\{x_1\} \subset \{x_1, x_2\}$. Note that shorthand notation is used where $g(1, 3)$ is equivalent to $g(\{x_1, x_3\})$ [99].

sources for a particular π-permutation. Aggregation of each permutation of inputs from these three sources can be represented as a path through the FM lattice. For example, consider a data sample x and evidence h that give rise to the permutation $\pi = \{2, 1, 3\}$. The corresponding FM values we use in the aggregation operation are $g(2)$, $g(2, 1)$, and $g(2, 1, 3)$, in that order. Thus, the aggregation of this instance of data can be visualized as the path shown in Fig. 1.2. This visualization strategy allows us to summarize the FM as well as the Choquet FI's paths. The lattice shown in Fig. 1.3 is the learned FM and paths for the training data from the wine data set using $m = 5$. In this visualization, the lines between nodes indicate the data support associated with each path. For the nodes that do not have any lines passing through them, i.e., the unsupported nodes the learned values are driven by the constraints at (1.9).

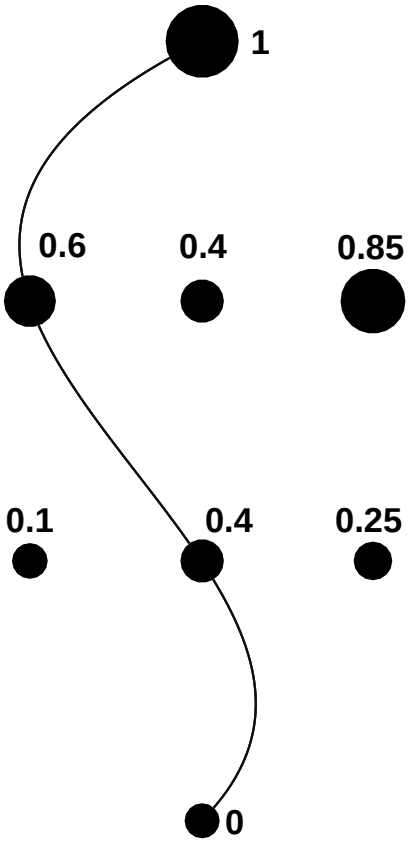

Figure 1.2: Path taken by the Choquet integral due to a single observation inducing the permutation $\pi = \{2, 1, 3\}$. Note that the FM was arbitrarily defined in this example, and their distribution (ordering) follows that of Fig. 1.1.

1.3 The DeFIMKL Algorithm

The DeFIMKL algorithm is a method of decision level fusion introduced in [97]. This algorithm uses the Choquet FI to non-linearly fuse the decisions from an ensemble of classifiers. Following is the mathematical description of the algorithm. Let $\mathbf{x}_i$ be an input feature vector and let $f_k(\mathbf{x}_i)$ be the decision-value produced by the kth classifier in an ensemble; the set of decisions made by the ensemble comprise the evidence h for the Choquet

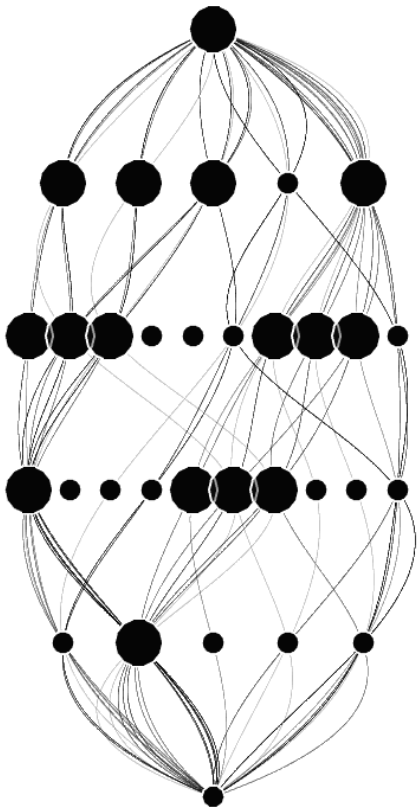

Figure 1.3: Lattice of learned FM and paths for the training data from the wine data set [33] using $m = 5$. Note that there are a few unsupported nodes and their learned values are driven by the constraints at (1.9). The arrangement of the nodes follows that of Fig. 1.1, where each level in the lattice represents all subsets of equal cardinality [99].

integral. The relative worth of each classifier is encoded in the FM, g, which is used by the Choquet FI to integrate the evidence. The result is the ensemble decision $C_g(\mathbf{x}_i)$ for feature-vector $\mathbf{x}_i$ with respect to the FM g,

$$C_g(\mathbf{x}_i) = \sum_{k=1}^{m} f_{\pi(k)}(\mathbf{x}_i) \left[g(A_k) - g(A_{k-1}) \right], \tag{1.2}$$

where $A_k = \{f_{\pi(1)}(\mathbf{x}_i), \ldots, f_{\pi(k)}(\mathbf{x}_i)\}$, such that $f_{\pi(1)}(\mathbf{x}_i) \geq f_{\pi(2)}(\mathbf{x}_i) \geq \ldots \geq f_{\pi(m)}(\mathbf{x}_i)$. Many previous works [10, 113, 125, 137] have explored this method as a generalized fusion approach for classifiers.

The behavior of the Choquet FI is solely defined by the FM, thus specifying an appropriate FM for a fusion problem is a key challenge. We employ a data-supported method to learn the FM, g, through regularized *sum-of-squared error* (SSE) optimization [4]. This method is summarized next.

Let the SSE be defined as

$$E^2 = \sum_{i=1}^{n} \left(C_g(\mathbf{x}_i) - y_i\right)^2. \tag{1.3}$$

It can be shown that (1.2), as a Choquet integral, can be reformulated as

$$C_g(\mathbf{x}_i) = \sum_{k=1}^{m} \left[f_{\pi(k)}(\mathbf{x}_i) - f_{\pi(k+1)}(\mathbf{x}_i)\right] g(A_k), \tag{1.4}$$

where $f_{\pi(m+1)} = 0$ [111]. We can then expand the SSE as

$$E^2 = \sum_{i=1}^{n} \left(H_{\mathbf{x}_i}^T \mathbf{u} - y_i\right)^2, \tag{1.5a}$$

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

$$H_{\mathbf{x}_i} = \begin{pmatrix} \vdots \\ f_{\pi(1)}(\mathbf{x}_i) - f_{\pi(2)}(\mathbf{x}_i) \\ \vdots \\ 0 \\ \vdots \\ f_{\pi(m)}(\mathbf{x}_i) - 0 \end{pmatrix}, \tag{1.5b}$$

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 $\mathbf{u}$. Note that the vector $H_{\mathbf{x}_i}$ when multiplied with $\mathbf{u}$, maps the difference terms in $H_{\mathbf{x}_i}$ with the appropriate FM values in $\mathbf{u}$; i.e., for each instance of data fusion, depending on the sort order, the m difference terms in (1.4) take the appropriate positions in the vector $H_{\mathbf{x}_i}$ while the rest remain as zero. There are only m non-zero terms in the vector $H_{\mathbf{x}_i}$. Finally, folding out the squared term in (1.5a) produces

$$\begin{aligned} E^2 &= \sum_{i=1}^{n} \left(\mathbf{u}^T H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T \mathbf{u} - 2 y_i H_{\mathbf{x}_i}^T \mathbf{u} + y_i^2 \right) \\ &= \mathbf{u}^T D \mathbf{u} + \mathbf{f}^T \mathbf{u} + \sum_{i=1}^{n} y_i^2, \\ D &= \sum_{i=1}^{n} H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T, \quad \mathbf{f} = -\sum_{i=1}^{n} 2 y_i H_{\mathbf{x}_i}. \end{aligned} \tag{1.6}$$

Since (1.6) is a quadratic function, we can add constraints on $\mathbf{u}$ such that it represents an 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

$$C = \begin{bmatrix} \Psi_1^T & \Psi_2^T & \cdots & \Psi_{n+1}^T & \cdots & \Psi_{m(2^{m-1}-1)}^T \end{bmatrix}^T \tag{1.7}$$

and Ψ_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

$$C = \begin{bmatrix} 1 & 0 & \cdots & -1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & -1 \end{bmatrix}. \tag{1.8}$$

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

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u}, \quad C\mathbf{u} \leq \mathbf{0}, \quad (0,1)^T \leq \mathbf{u} \leq \mathbf{1}, \tag{1.9}$$

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

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda v_*(\mathbf{u}), \tag{1.10}$$

where λ is the regularization weight and $v_*(\cdot)$ is some regularization function. For example, ℓ_p-norm regularization is applied when $v_*(\mathbf{u}) = \|\mathbf{u}\|_p$. ℓ_1 and ℓ_2 regularization of this QP were discussed in [4, 97].

The QPs at (1.9) and (1.10) 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 (1.2).

1.4 Challenges with ChI-based aggregation methods

ChI is a powerful aggregation operator capable of modeling the worth of all possible subsets of input sources. However, the number of FM parameters for ChI scales as 2^m where m is the number of input sources. Training so many FM parameters requires proportionally large amount of training data, or this could lead to over-fit solutions. This work presents novel regularization strategies to address the over-fitting problem of ChI-based data fusion. The proposed goal-based regularization strategies allow the user to encode knowledge of the underlying FM to the learning problem.

The boundary and monotonicity constraints of FM have limited the applicability of ChI-based aggregation to decision-level fusion. These constrains on the FM were relaxed in [59] to introduce a ChI-based regression method, *choquet integral regression* (CIR). This work enhances CIR by applying ℓ_2-regularization to alleviate the problem of over-fitting. This work also introduces visualization techniques and metrics to enhance the explainability of the learned regression model as well as the model outputs.

As the number of FM parameters scales exponentially with the number of input sources,

the exhorbitant space complexity associated with the QP-based DeFIMKL algorithm has limited the practical applicability of ChI-based aggregation to six or fewer input sources. This work introduces an online learning method that addresses this limitation of the ChI. The online method is a gradient descent approach that iteratively processes the training data, one data point at a time, and therefore requires significantly less computation and space at any time during the training. The online method enabled the applicability of ChI-based data fusion methods on higher dimensional data sets.

1.5 Dissertation Outline and Contributions

This section outlines my work on feature- and decision-level fusion problems. A high-level summary on each chapter along with corresponding novel contributions is presented.

Chapter 2 presents a novel aggregation method called the *extended linear order statistic* (ELOS). Unlike the simpler aggregation method, the *linear order statistic* (LOS), which parametrizes the aggregation of d features with d regression weights, the ELOS learns d parameters for each position in the sorted input vector, one for each input feature; thus, ELOS learns a total of d^2 weights for the aggregation of d features. The increased number of parameters helps the algorithm improve its expressibility while maintaining interpretability.

Chapter 3 presents the CIR, and introduces a novel method to apply ℓ_2-regularization on

the CIR training algorithm. Application of ℓ_2-regularization has resulted in statistically significant improvement in the performance of the algorithm in 73% of the experiments. The visualization and analysis tools presented in this chapter further enhance the explainability of the CIR models.

Chapter 4 addresses the limitation of the CIR models presented in Chapter 3 due to its exorbitant space complexity. The online method introduced in this chapter applies an iterative gradient descent method to train the CIR. This method processes one observation at a time, enabling the applicability of CIR on higher dimensional data sets.

Chapter 5 reviews the data support-based training approaches for ChI-based decision fusion, and introduces several novel regularization strategies to address the problem of overfitting. The goal-based regularization strategies presented in this chapter allow the user to consciously push the learned FM towards a predefined structure; these regularization methods allow the user to encode knowledge or intuition of the underlying FM to the learning problem. This chapter also proposes a novel approach that incorporates the data support of the FM variables and the corresponding degree of disparity between sources (the relative support across all variables) in the learning algorithm. In this approach, the amount of regularization is directly related to the degree of support, i.e.,variables with a strong data-support will be regularized to a very low degree, while variables with limited support will be regularized to a greater extent. This method has shown to improve the model quality by explicitly considering the uncertainty due to limited training data volume and variety.

Chapter 6 introduces an online algorithm for training ChI-based decision fusion. This iterative algorithm, in addition to minimizing the SSE, also ensures that the monotonicity and boundary conditions of the FM are maintained. The online algorithm introduced in this chapter enables the extension of the applicability of ChI-based decision fusion to higher dimensional data sets.

Chapter 7 concludes this dissertation and discusses future work.

1.6 List of Relevant Publications

The work collected in this dissertation is based on the following publications:

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (July, 2020). "Extended Linear Order Statistic," *IEEE International Conference on Fuzzy Systems*.

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (July, 2020). "Choquet Integral Ridge Regression," *IEEE International Conference on Fuzzy Systems.*

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (Dec 2020). "Visualization and Analysis Tools for Explainable Choquet Integral Regression," *IEEE Symposium Series on Computational Intelligence.*

S.K. Kakula, A.J. Pinar, Muhammed A. Islam, D.T. Anderson, and T.C. Havens (2020).

"Novel Regularization for Learning the Fuzzy Choquet Integral with Limited Training Data," *IEEE Transactions on Fuzzy Systems.*

S.K. Kakula, T.C. Havens, A.J. Pinar, Muhammed A. Islam, D.T. Anderson, Andrew Buck, and Tim Wilkin. "Learning the Fuzzy Choquet Integral in the Presence of Uncertainty Caused by Limited Training Data Volume and Variety" Under Review - *IEEE International Conference on Fuzzy Systems.*

S.K. Kakula, A.J. Pinar, D.T. Anderson, and T.C. Havens (Sep 2020). "Online Learning of the Fuzzy Choquet Integral," *IEEE International Conference on Systems, Man, and Cybernetics (SMC).*

S.K. Kakula, T.C. Havens, and D.T. Anderson. "Online Sequential Learning of Monotonic Fuzzy Measures in the Choquet Integral," under review, *IEEE International Conference on Fuzzy Systems.*

Chapter 2

Extended Linear Order Statistic (ELOS) Aggregation and Regression

2.1 Introduction

The *ordered weighted average* (OWA) aggregation operator was introduced by Yager in 1988 [127]. It was primarily designed to aggregate the outputs from multiple decision makers to produce an overall fused decision function. An OWA operator on d dimensional data is a mapping $F : \mathcal{R}^d \rightarrow \mathcal{R}$. Given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and the

corresponding weight vector $\mathbf{v}$, the OWA function is given by

$$\text{OWA}(\mathbf{x}, \mathbf{v}) = \sum_{i=1}^{d} v_i x_{(i)}, \tag{2.1}$$

where $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(d)}$, $v_i \geq 0$, and $\sum_{i=1}^{d} v_i = 1$.

The OWA induces non-linearity in the solution by sorting the input vector prior to the aggregation operation. It also limits the outputs of aggregation between the minimum and the maximum values of the input sample $\mathbf{x}$, and thus is best suited for decision-level fusion. In Yager's later work [129], he extended the application of OWA to regression problems. This work introduced an OWA-based approach to evaluate the fitness of a solution to the data, where, the weighting vector of the OWA operator controls the penalties for each data point, based on the magnitude of the error measure (e.g. squared-error). Yager *et al.* demonstrated that OWA-based regression provides a generic formulation of the regression problem in which existing classical methods like *least squares* (LS) regression, *least absolute deviation* (LAD) regression, and *maximum likelihood* (ML) estimators are special cases. Also, the OWA-based regression solutions were found to be less sensitive to outliers as compared to the traditional methods like LS, LAD, and ML-estimators.

The OWA function at (2.1) can be modified to

$$\text{OWA}_g(\mathbf{x}, \mathbf{v}) = \frac{\sum_{i=1}^{d} v_i x_{(i)}}{\sum_{i=1}^{d} v_i}, \tag{2.2}$$

where $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(d)}$, and $v_i \geq 0$. While this is equivalent to (2.1), as it implicitly encodes the constraint on the weights on $\mathbf{x}$ to sum to 1, it does help with certain learning problems. This form can be relaxed to the *linear order statistic* (LOS), which has the form

$$\text{LOS}(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{d} w_i x_{(i)}, \tag{2.3}$$

where the weights $\mathbf{w}$ are no longer constrained to sum to 1, and can also take negative values. This enables the aggregation operation to behave more like a regression operator that can map the input to any value on the set of reals.

An LOS for the aggregation of d sources is parameterized by d values, each representing the weight corresponding to each position in the sorted input vector, $\mathbf{x}_\pi = (x_{(1)}, x_{(2)}, \ldots, x_{(d)})$. While having just d parameters makes the solution more explainable, since we have only a single parameter for each sorted position, the LOS algorithm is quite limited in terms of the amount of non-linear space it explores for an optimal solution—i.e., its "expressibility" is limited. We propose a novel aggregation method called the *Extended Linear Order Statistic* (ELOS), where the aggregation of d sources is parameterized by d^2 weights. For each position in the sorted input vector we again have d weights, one for each source. The increased number of parameters helps the algorithm improve its *expressibility*, but it still maintains its *interpretability*.

The remainder of this chapter is organized as follows. Section 2.2 presents the background

on OWA operators and OWA-based regression, then Section 2.3 discusses the problem formulation and training process of LOS. In Section 2.4, we introduce ELOS and describe the training process. Section 2.5 discusses the ℓ_1- and ℓ_2-regularization. We then compare the performance of ELOS with linear regression and LOS in Section 2.6. Section 2.7 summarizes this work and discusses possible future work.

2.2 Background

The OWA has been used in many fields, such as decision making [88, 89, 104, 118], risk analysis [38, 105], environment assessment [103, 134], and sports performance analysis [2, 108]. Given the wide range of applications, several OWA-based aggregation operators were proposed. *Induced ordered weighted average* (IOWA) by Yager *et al.* [130] introduced a modified ordering approach where the ordering is induced by a variable called the order inducing variable. Chiclana *et al.* [26] introduced the *ordered weighted geometric* (OWG) aggregation operator, a geometric mean-based OWA operator. Yager *et al.* [128] introduced *continuous* OWA (C-OWA) to aggregate continuous interval values. While most of these developments were oriented towards OWA-based aggregation tools, in 2009, Yager *et al.* [129] extended the application of OWA to regression problems and demonstrated that OWA-based regression particularly outperforms traditional least-squares

and least-absolute-deviation methods when the data contains a significant portion of outliers. The ELOS regression approach we propose builds on these prior works and parameterizes the aggregation of d-dimensional inputs using d^2 parameters, wherein for each of the d sorted positions in the input we again have d weights, each corresponding to individual variables in the input vector—more details in Section 2.4.

2.3 Problem Formulation

Given a set of training data $(\mathbf{y}, X)$, where $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \subset \mathbb{R}^d$ (a set of feature vectors) and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ (a vector of outputs)[1], the classic regression problem involves learning a function that maps the input data X to the output. Such function is a parameterized model such that

$$\mathbf{y} \approx f(\mathbf{x}, \mathbf{w}),$$

where $\mathbf{w}$ is the set of learned parameters of the regression function f. During training, the regression parameters $\mathbf{w}$ are optimized with respect to an error function, usually squared-error,

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(f(\mathbf{x}_i, \mathbf{w}) - \mathbf{y}_i\right)^2. \tag{2.4}$$

[1]Note that the output $\mathbf{y}$ could be extended to multiple outputs for each input vector without loss of generality.

Consider the prepended input $\mathbf{x}_i = (x_{i,0}, x_{i,1}, x_{i,2}, \ldots x_{i,d})^T$, where $x_{i,0}$ is defined as the constant bias multiplier 1, and $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})^T$ are the d-features of the input, the function $f(\mathbf{x}_i, \mathbf{w})$ takes the form

$$f(\mathbf{x}_i, \mathbf{w}) = \sum_{j=0}^{d} x_{i,j} w_j = \mathbf{w}^T \mathbf{x}_i, \tag{2.5}$$

where w_o is the bias term and each weight in $(w_1, w_2, \ldots, w_d)^T$ is the coefficient of the corresponding variable in the input vector $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$. This is the well-known least-squares problem with a closed-form solution for (2.4),

$$\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}, \tag{2.6}$$

where

$$X^T = \begin{bmatrix} 1 & \mathbf{x}_1^T \\ 1 & \mathbf{x}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^T \end{bmatrix} \tag{2.7}$$

is the $n \times (d+1)$ input matrix in which each row is an input vector (with the prepended bias multiplier 1 in the first position), and $\mathbf{y}$ is the vector of outputs in the training set. For more extensive details on regression, in general, we suggest [58].

2.3.1 Linear Order Statistic (LOS) Regression

The regression function for LOS takes the same form as (2.5) except that the input vectors $\mathbf{x}_i$ are first sorted in descending order,

$$f_{LOS}(\mathbf{x}_i, \mathbf{w}) = \sum_{j=0}^{d} (\mathbf{x}_i)_{\pi_i(j)} w_j = \mathbf{w}^T (\mathbf{x}_i)_{\pi_i}, \tag{2.8}$$

where π is a sorting function, such that $(\mathbf{x}_i)_{\pi_i(1)} \geq (\mathbf{x}_i)_{\pi_i(2)} \geq \cdots \geq (\mathbf{x}_i)_{\pi_i(d)}$; $(\mathbf{x}_i)_{\pi_i(0)} = 1$ is defined so that w_0 represents the bias in the regression. Thus, w_1 corresponds to the weight on the input variable with the highest magnitude, w_2 corresponds to the weight on the next highest variable, and so on. The closed-form solution at (2.6) also applies to LOS-regression by simply forming the following sorted input data matrix,

$$X_\pi^T = \begin{bmatrix} 1 & (\mathbf{x}_1)_{\pi_1}^T \\ 1 & (\mathbf{x}_2)_{\pi_2}^T \\ \vdots & \vdots \\ 1 & (\mathbf{x}_n)_{\pi_n}^T \end{bmatrix}, \tag{2.9}$$

where $(\mathbf{x}_i)_{\pi_i}$ is simply the sorted version of the ith input vector. Finally, the LOS weight vector $\mathbf{w}$ that minimizes (2.4) can be calculated by

$$\mathbf{w}^* = ({X_\pi}^T X_\pi)^{-1} {X_\pi}^T \mathbf{y}. \tag{2.10}$$

2.4 Extended Linear Order Statistic

While the LOS-regression solution of a d-dimensional input comprises one weight each for each position in the sorted input and an additional bias parameter, ELOS trains d weights for each position in the sorted input, where each weight corresponds to an individual variable, plus a bias parameter; i.e., the regression solution comprises $d^2 + 1$ weights.

Again, consider an input vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})^T$, where $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ are the d-features of the input. The regression function for ELOS takes the form

$$f_{ELOS}(\mathbf{x}_i, \mathbf{w}) = \sum_{j=1}^{d} (\mathbf{x}_i)_{\pi_i(j)} w_{j,i} + \beta, \tag{2.11}$$

where π is again a sorting function, such that $(\mathbf{x}_i)_{\pi_i(1)} \geq (\mathbf{x}_i)_{\pi_i(2)} \geq \cdots \geq (\mathbf{x}_i)_{\pi_i(d)}$, and here β is the bias term. The regression weights are now a d^2 matrix, as shown in Table 2.1. We first write the ELOS regression in this way for ease of understanding, but later we will extend this formulation for ease of data-driven learning of W and β.

A graphical representation of the associated weights for each input is shown in Table 2.1 for a 5-dimensional input vector, $\mathbf{x} = (1.3, 0.7, -0.2, 2.1, 1.6)^T$. The sorting function on this vector would be $\pi = (4, 5, 1, 2, 3)$; hence, the bold weights in the shown matrix would be the weights applied to this input vector. Essentially, the ELOS combines the power of linear regression with that of the LOS regression; each row of W is associated with each

Table 2.1
ELOS Weight Matrix for 5-dimensional Data

Input	Sort Order, $\pi(i)$				
$x_{i,1}$	$w_{1,1}$	$w_{1,2}$	$\boldsymbol{w_{1,3}}$	$w_{1,4}$	$w_{1,5}$
$x_{i,2}$	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$\boldsymbol{w_{2,4}}$	$w_{2,5}$
$x_{i,3}$	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$	$\boldsymbol{w_{3,5}}$
$x_{i,4}$	$\boldsymbol{w_{4,1}}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$
$x_{i,5}$	$w_{5,1}$	$\boldsymbol{w_{5,2}}$	$w_{5,3}$	$w_{5,4}$	$w_{5,5}$

5×5 weight matrix W for ELOS regression of 5-dimensional data. The weights selected for the aggregation of the example input vector $\mathbf{x} = (1.3, 0.7, -0.2, 2.1, 1.6)^T$ are marked in bold font. Note that we will learn an additional bias weight β, for a total of $5^2 + 1 = 26$ parameters.

element of the input vector (like linear regression), and each column of W corresponds to the sort of the input elements (like LOS regression).

The ELOS formulation at (2.11) is good for illustrating how ELOS works, but this is problematic for data-driven learning of W and β. Hence, we reform the input vectors $\mathbf{x}$ and the weight matrix W as follows. It may help to examine Fig. 2.1 as you read along with the following mathematical explanation. First, consider the extension of the ith input vector $\mathbf{x}_i$,

$$\mathbf{x}_i^e = (1, (\mathbf{x}_i)_{\pi_i}^e)^T, \tag{2.12}$$

where the first element of 1 is included so that the bias can be implicitly included in $\mathbf{w}^e$ (which we describe later). The vector $(\mathbf{x}_i)_{\pi_i}^e$ is a d^2-length vector with only d non-zero terms; this vector will enforce the sort, as indicated by π. The vector $(\mathbf{x}_i)_{\pi_i}^e$ has the form

$$(\mathbf{x}_i)_{\pi_i}^e = ([(x_{i,1})_{\pi_i}^e]^T, [(x_{i,2})_{\pi_i}^e]^T, \ldots, [(x_{i,d})_{\pi_i}^e]^T)^T. \tag{2.13}$$

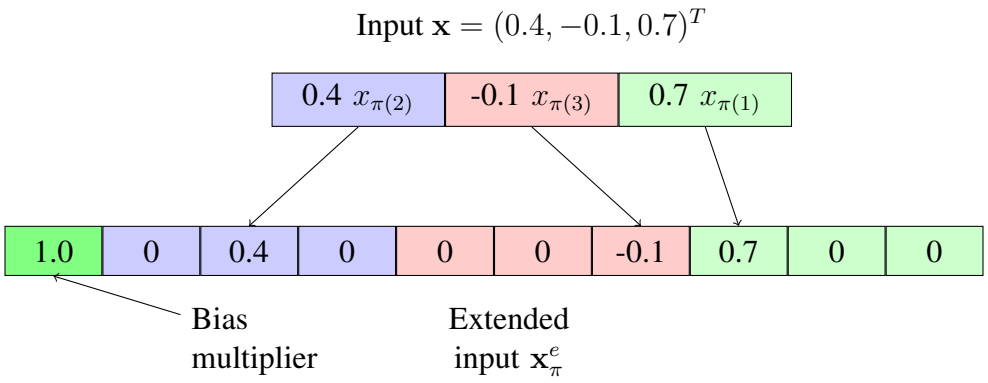

Figure 2.1: Transformation of an example 3-dimensional input vector $\mathbf{x}$ to its extended form.

Each of $[(x_{i,j})^e_{\pi_i}]$ is simply a vector of zeroes, with each element of $\mathbf{x}_i$ sorted into the corresponding spot in the sort. Figure 2.1 illustrates the construction of $(\mathbf{x})^e_\pi$ for the example input vector $\mathbf{x} = (0.4, -0.1, 0.7)^T$, with sort $\pi(1) = 3, \pi(2) = 1, \pi(3) = 2$. The first element of $(\mathbf{x})^e_\pi$ is the bias multiplier. The blue chunk is $(x_1)^e_\pi$, where x_1 has been sorted into the second spot. The red chunk is $(x_2)^e_\pi$, where x_2 has been sorted into the third spot. And similarity for the green chunk, where x_3 has been sorted into the first spot. While this may seem complicated notation, it significantly simplifies the data-driven learning process. Since each input vector $\mathbf{x}_i$ is extended to $(\mathbf{x}_i)^e_{\pi_i}$, the weight matrix W must be correspondingly extended.

Let the extended form of W be

$$\mathbf{w}^e = (\beta, w_{1,1}, w_{1,2}, \ldots, w_{1,d}, w_{2,1}, \ldots, w_{d,d})^T, \tag{2.14}$$

where all we have done is take each row of W sequentially to form a long vector and

prepended the bias term as the first element of $\mathbf{w}^e$.

We can now rewrite (2.11) as

$$f_{ELOS}(\mathbf{x}_i, \mathbf{w}) = (\mathbf{w}^e)^T \mathbf{x}_i^e, \tag{2.15}$$

which you can see is most pleasing—we have essentially written ELOS regression as a linear regression equation.

Finally, it is easy to see that ELOS can be solved much the same as linear and LOS regression were solved,

$$(\mathbf{w}^e)^* = ([X^e]^T X^e)^{-1} [X^e]^T \mathbf{y}, \tag{2.16}$$

where

$$[X^e]^T = \begin{bmatrix} (\mathbf{x}_1^e)^T \\ (\mathbf{x}_2^e)^T \\ \vdots \\ (\mathbf{x}_n^e)^T \end{bmatrix}, \tag{2.17}$$

Once the (d^2+1)-ELOS weight vector $(\mathbf{w}^e)^*$ is learned using a training data set, the ELOS regression output for a new input $\mathbf{x}$ can be calculated using (2.15). Example 1 demonstrates the ELOS regression calculation in more detail.

Example 1. Consider the problem of learning an ELOS regression model on a 5-dimensional training data set $(\mathbf{y}, X)$, where $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, $\mathbf{x}_i \subset \mathbb{R}^5$ and $\mathbf{y} = (y_1, y_2, ..., y_n)^T$. Consider the input vector $\mathbf{x} = (1.3, 0.7, -0.2, 2.1, 1.6)^T$. The sort order of the variables in $\mathbf{x}$ are $\pi(1) = 4, \pi(2) = 5, \pi(3) = 1, \pi(4) = 2, \pi(5) = 3$. Thus, the weight applied to the fourth input x_4 would be $w_{4,1}$, the weight applied to the fifth element x_5 is $w_{5,2}$, and so on as shown in Table 2.1. Thus the output is calculated as

$$y = \beta + w_{4,1}x_4 + w_{5,2}x_5 + w_{1,3}x_1 + w_{2,4}x_2 + w_{3,5}x_3. \tag{2.18}$$

Remark 1. It is easy to show that ELOS is equivalent to linear regression or LOS regression when the weight matrix W takes a certain form. ELOS is equivalent to linear regression if the rows of W, illustrated in Table 2.1, are constant-valued. That is if $w_{i,1} = w_{i,2} = \ldots = w_{i,d}, \forall i$.

Similarly, ELOS is equivalent to LOS regression if the columns are equal: $w_{1,j} = w_{2,j} = \ldots, w_{d,j}, \forall j$.

This Remark illustrates that ELOS can do everything both linear and LOS regression are able to do. The only concern is whether ELOS will over-fit to training data. We now turn to describing how we can apply regularization to the regression methods described in this chapter.

2.5 Regularization

While increasing the number of learned parameters might improve the expressibility of the algorithm, more parameters may sometimes capture the noise in the training data and thereby result in an over-fit solution. Regularization allows us to restrict the size of the learned parameters and thus discourages the algorithm from learning a solution that is more complex than necessary. Experiments on ELOS and comparable regression methods presented in Section 2.6 explored the impact of ℓ_1- and ℓ_2-regularization.

2.5.1 ℓ_2-regularization: Ridge regression

The SSE function at (2.4) can be modified to include the ℓ_2-regularization penalty to make the ℓ_2-penalized-SSE function

$$SSE_{\ell_2} = \sum_{i=1}^{n} \left(f(\mathbf{x}_i, \mathbf{w}) - y_i\right)^2 + \lambda \sum_{j=1}^{d} w_j^2,\ \lambda \geq 0, \tag{2.19}$$

where λ is the regularization parameter. Each of the regressions (linear, LOS, and ELOS) at (2.5), (2.8), and (2.15) can be written in the form of a simple dot-product $\mathbf{w}^T\mathbf{x}$; hence,

(2.19) can be rewritten as

$$SSE_{\ell_2} = \sum_{i=1}^{n} \left(\mathbf{w}^T \mathbf{x}_i - y_i\right)^2 + \lambda \sum_{j=1}^{d} w_j^2. \tag{2.20}$$

Expanding (2.20) gives

$$SSE_{\ell_2} = \left(\mathbf{w}^T X - \mathbf{y}\right)^T \left(\mathbf{w}^T X - \mathbf{y}\right) + \lambda \|\mathbf{w}\|_2^2. \tag{2.21}$$

By taking the derivative of (2.21) and setting to zero, it can be shown that SSE_{ℓ_2} is minimized when

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}, \tag{2.22}$$

which is the well-known ridge-regression solution. While (2.22) is notated for linear regression, this can be applied to both LOS and ELOS by replacing X with X_π or X^e and the appropriate form of the weight vector $\mathbf{w}$. For more extensive detail on ℓ_2-regularization, in general, we suggest [58]. We used the Matlab's `fitrlinear` function to apply ℓ_2-regularization, which accounts for numerical issues that can occur with the closed-form solution at (2.22).

2.5.2 ℓ_1-regularization: Lasso regression

The SSE function at (2.4) can be modified to include the ℓ_1-regularization penalty as

$$SSE_{\ell_1} = \sum_{i=1}^{n} \left(f(\mathbf{x}_i, \mathbf{w}) - \mathbf{y}_i\right)^2 + \lambda \sum_{j=1}^{d} |\mathbf{w}_j|, \tag{2.23}$$

where λ is again the regularization parameter. Unlike ridge regression, ℓ_1-regularization does not have a closed-form solution. We used Matlab's `fitrlinear` function to apply ℓ_1-regularization. Matlab implements the *Alternating Direction Method of Multipliers* (ADMM) algorithm [16] to solve for the optimal weight vector $\mathbf{w}$ subject to ℓ_1 regularization.

2.6 Experiments

We tested the ELOS algorithm on real world data sets from the UCI machine learning repository [33]. Using *mean squared error* (MSE) as the performance measure, we compared ELOS with linear regression and LOS regression on 10 benchmark data sets[2]. We also evaluated the impact of ℓ_1- and ℓ_2-regularization on each of these methods through a grid search over a set of values for the regularization parameter λ, ranging on a logarithmic scale between 0.0001 and 1000. We reported the results with the best λ. Each experiment

[2]See Table A.2 in Appendix A for details on the UCI regression data sets used in the experiments.

consisted of 100 randomized trials, where the result of each trial is the average MSE calculated over a 10-fold cross validation. Table 4.2 presents the experimental results, where the MSE reported in each cell is the average MSE of 100 experimental trials; its standard deviation is presented in parentheses. All the experiments are implemented in Matlab.

2.6.1 ELOS versus linear regression

ELOS, unlike linear regression, learns a weight vector for each feature in the training data—one for each sort position. Figures 2.2 and 2.3 compare the weights learned by ELOS and linear regression on Airfoil and Concrete data sets, respectively. In both these figures, we see that the ELOS weights for each feature are spread on either side of the linear regression weights, thus allowing ELOS to treat the features differently depending on their sort order. These figures show that the overall values of the weights of ELOS follow that of the linear regression weights, which is intuitively pleasing.

2.6.2 ELOS vs. LOS

Figures 2.4 and 2.5 compare the learned parameters for ELOS and LOS on the Airfoil and Concrete data sets, respectively. While the LOS has learned one weight for each sorted position, ELOS learns a weight vector for each feature and applies weights according to the

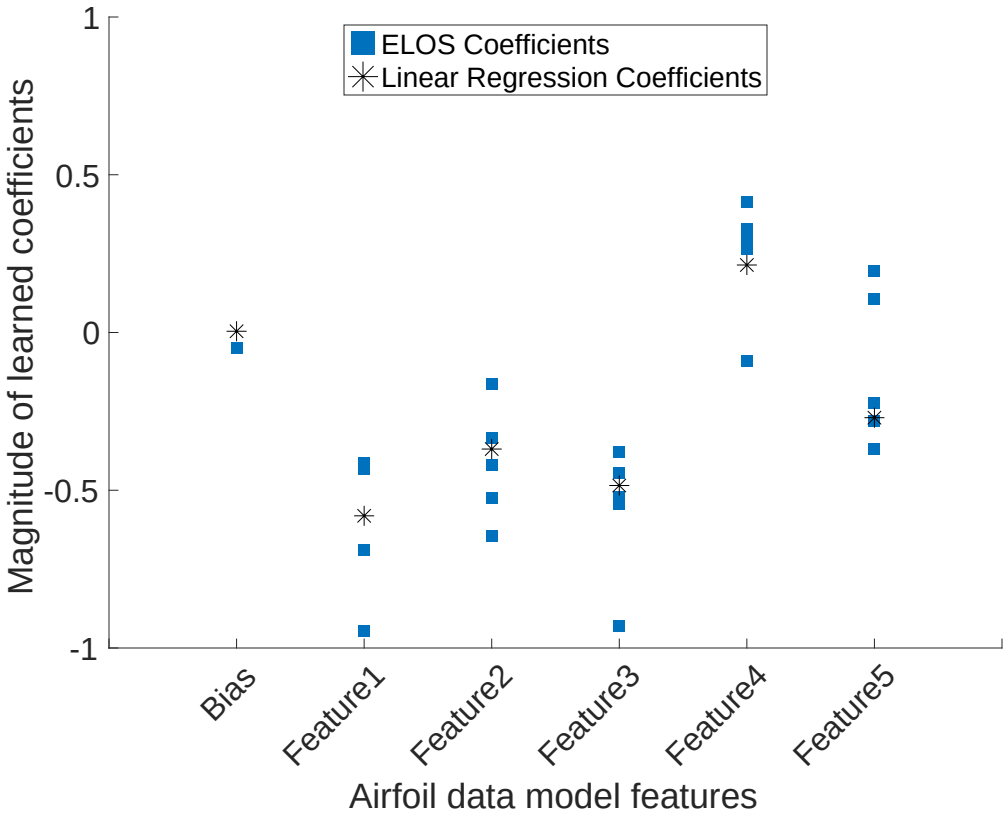

Figure 2.2: Comparison of learned parameters of ELOS and linear regression on Airfoil data set. For each feature, ELOS has learned 5 weights, each corresponding to sort position of that feature, whereas linear regression learns only one weight per feature. ELOS was able to capture non-linearity in the input-output relation, which is represented by the variation in the learned weights for each feature.

sort. In both Figures 2.4 and 2.5, the high variance of ELOS weights about the LOS weights for each feature demonstrate the flexibility of ELOS to treat each feature differently based on their sort position. Thus, linear regression and LOS are special cases within ELOS, since ELOS, in addition to learning the weights for individual features, also explores the non-linearity introduced by the sorting of input vector.

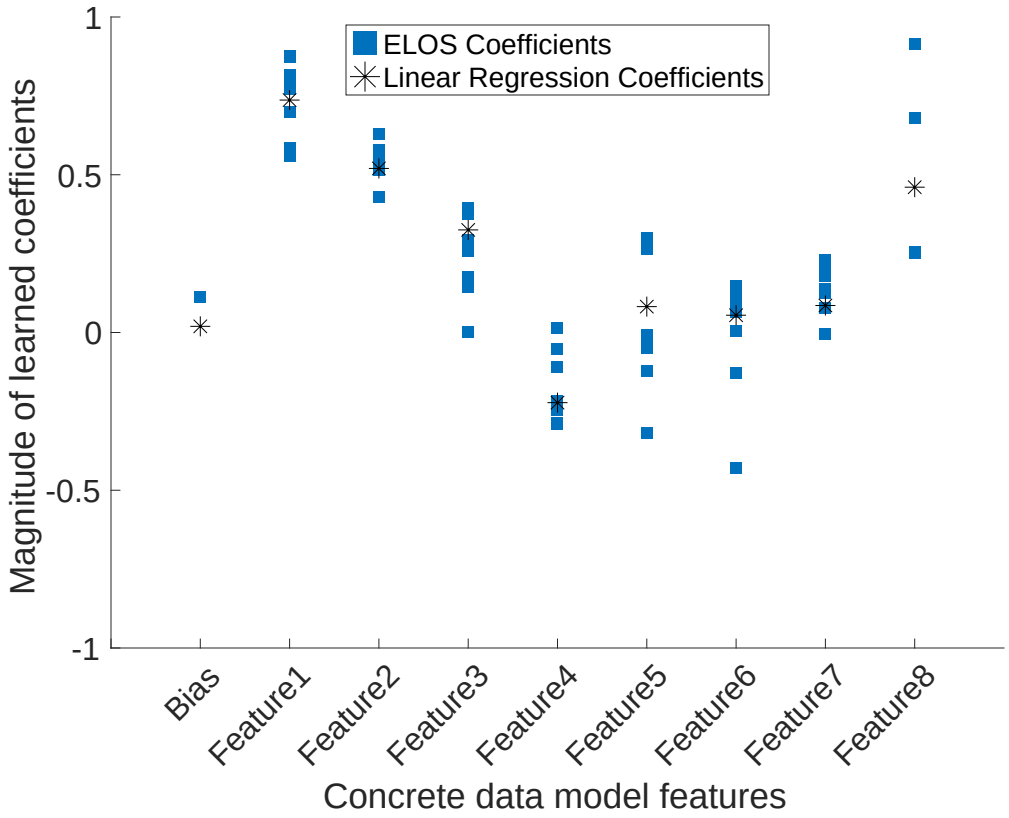

Figure 2.3: Comparison of learned parameters of ELOS and linear regression on Concrete data set. For each feature, ELOS has learned 8 weights, each corresponding to sort position of that feature.

2.6.3 Results on benchmark data sets

Table 4.2 shows the performance comparison of ELOS and the other competing methods on real-world data sets. The MSE values presented in the table are the average values taken over 100 randomized experimental trails, where the MSE of each trial is the mean MSE over a 10-fold cross validation. The best algorithms on each of these data sets were marked in bold font. We performed a two-sample t-test at a 5% significance level to determine

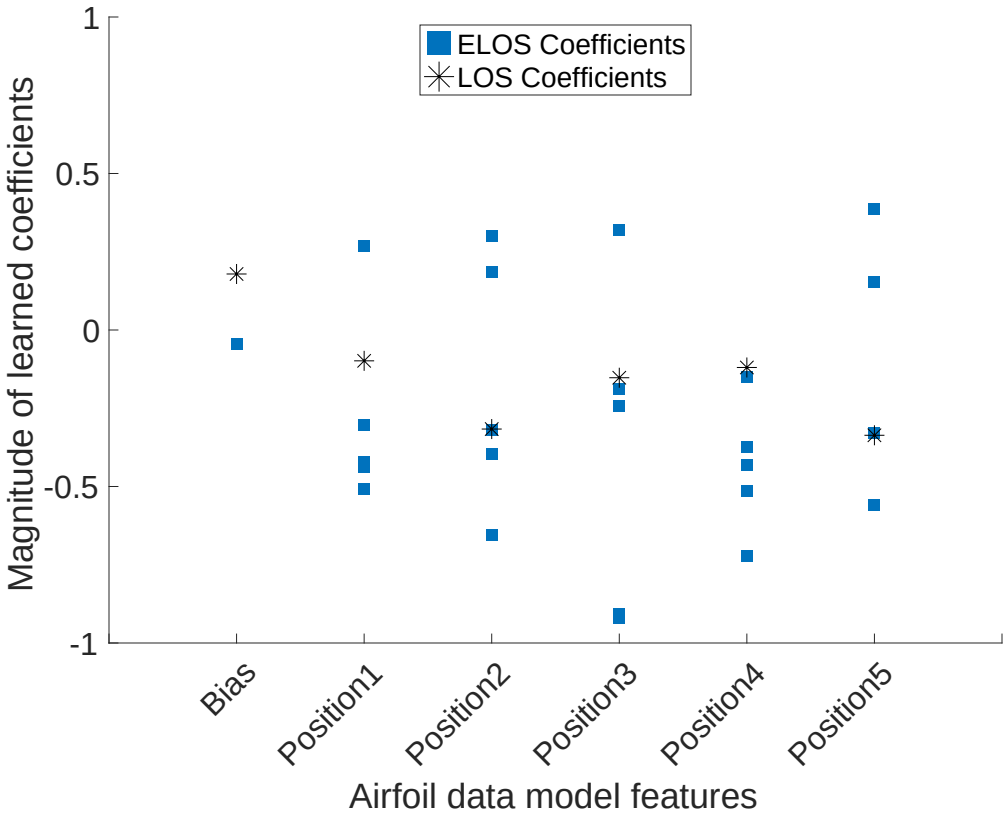

Figure 2.4: Comparison of learned parameters of ELOS and LOS on Airfoil data set. For each sort position, ELOS has learned 5 weights, one for each feature, where as the LOS has learned only one weight per sort position.

the statistically best results—hence, more than one algorithm can be considered as best. The last column in Table 4.2 shows the total number of data sets on which the algorithm produced the best results. Overall, ELOS performed better than Linear regression and LOS in 8 out 10 instances. Regularization did not seem to have a strong impact on ELOS since ℓ_1- and ℓ_2-regularized versions performed better than unregularized-ELOS only on two out of 10 data sets.

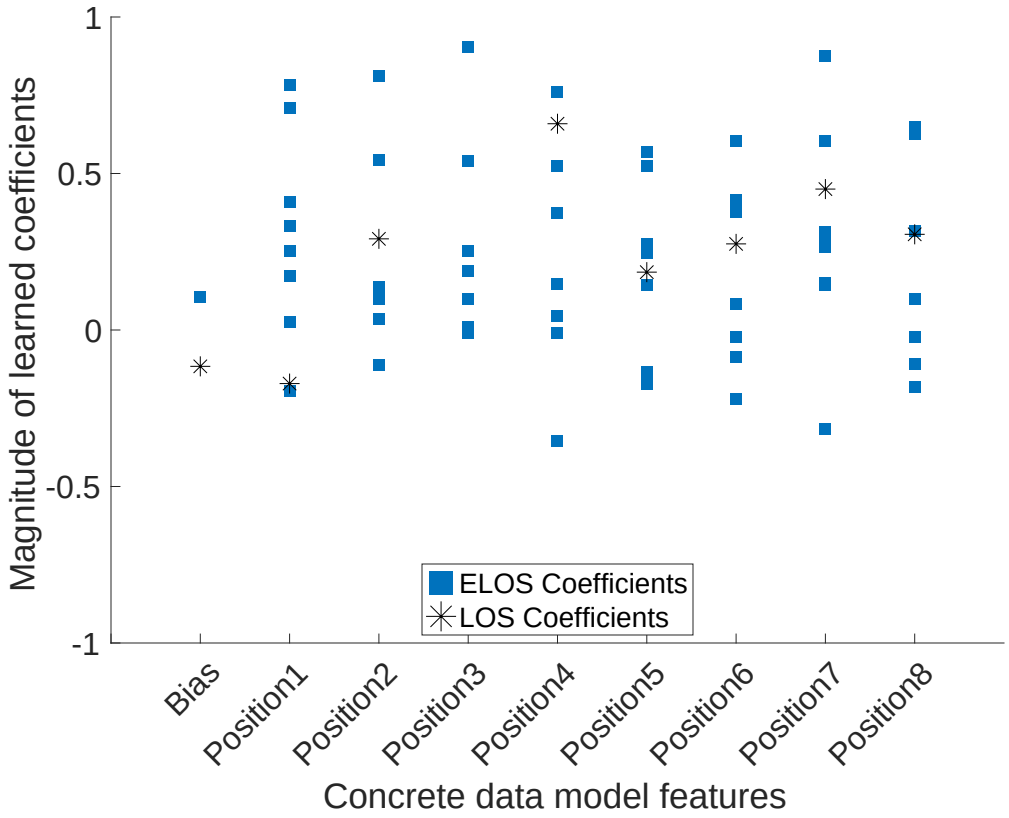

Figure 2.5: Comparison of learned parameters of ELOS and LOS on Concrete data set. For each sort position, ELOS has learned 8 weights, one for each feature, where as the LOS has learned only one weight per sort position

2.7 Conclusion and future work

This chapter introduced ELOS, an OWA-based regression operator and demonstrated that it is a significant improvement over simple linear and LOS regression. ELOS, by learning a weight vector for each input feature—one weight for each sort position—treats each variable independently and also enables the non-linearity introduced by the sorting process.

Table 2.2
MSE on Benchmark Data Sets

Method	Concrete	Real Estate	Fish Toxicity	Aquatic Toxicity	Red Wine	White Wine	ENB-2	Yacht	Airfoil	ISE	# of best instances
n	1,030	414	908	546	1599	4898	768	308	1503	536	-
d	8	5	6	8	11	11	8	6	5	7	-
ELOS	**0.345 (0.003)**	0.379 (0.006)	**0.403 (0.004)**	**0.558** (0.012)	**0.67 (0.003)**	**0.757 (0.001)**	**0.057 (0.001)**	**0.332 (0.011)**	**0.392 (0.002)**	0.506 (0.008)	7
ELOS-ℓ_1	0.359 (0.004) $\lambda = 0.01$	**0.377 (0.005)** $\lambda = 0.0001$	0.409 (0.004) $\lambda = 0.001$	0.561 (0.007) $\lambda = 0.01$	0.681 (0.003) $\lambda = 0.001$	0.772 (0.001) $\lambda = 0.001$	0.062 (0.001) $\lambda = 0.0001$	0.408 (0.01) $\lambda = 0.0001$	0.4 (0.002) $\lambda = 0.001$	0.496 (0.009) $\lambda = 0.001$	1
ELOS-ℓ_2	0.362 (0.004) $\lambda = 0.01$	**0.374 (0.005)** $\lambda = 0.01$	0.408 (0.004) $\lambda = 0.01$	0.567 (0.01) $\lambda = 0.01$	0.678 (0.004) $\lambda = 0.01$	0.773 (0.001) $\lambda = 0.001$	0.059 (0.001) $\lambda = 0.001$	0.398 (0.011) $\lambda = 0.0001$	0.401 (0.002) $\lambda = 0.001$	0.497 (0.009) $\lambda = 0.01$	1
Linear	0.424 (0.002)	0.445 (0.002)	0.437 (0.002)	**0.553** (0.003)	0.661 (0.001)	0.729 (0.001)	0.087 (0.000)	0.520 (0.003)	0.505 (0.001)	**0.437 (0.003)**	1
Linear-ℓ_1	0.418 (0.003) $\lambda = 0.01$	0.445 (0.002) $\lambda = 0.01$	0.435 (0.002) $\lambda = 0.01$	**0.550** (0.005) $\lambda = 0.01$	0.660 (0.001) $\lambda = 0.01$	0.728 (0.001) $\lambda = 0.01$	0.087 (0.001) $\lambda = 0.0001$	0.519 (0.004) $\lambda = 0.0001$	0.501 (0.001) $\lambda = 0.01$	**0.437 (0.003)** $\lambda = 0.001$	1
Linear-ℓ_2	0.411 (0.002) $\lambda = 0.1$	0.445 (0.002) $\lambda = 0.01$	0.434 (0.002) $\lambda = 0.1$	**0.552** (0.004) $\lambda = 0.1$	0.658 (0.001) $\lambda = 0.1$	0.728 (0.001) $\lambda = 0.1$	0.087 (0.001) $\lambda = 0.0001$	0.519 (0.004) $\lambda = 0.0001$	0.493 (0.001) $\lambda = 0.1$	**0.437 (0.003)** $\lambda = 0.001$	1
LOS	0.708 (0.002)	0.727 (0.005)	0.718 (0.003)	0.940 (0.007)	0.996 (0.002)	0.977 (0.001)	0.758 (0.003)	1.086 (0.009)	0.797 (0.001)	0.508 (0.008)	0
LOS-ℓ_1	0.707 (0.002) $\lambda = 0.01$	0.727 (0.007) $\lambda = 0.001$	0.709 (0.002) $\lambda = 0.01$	0.934 (0.007) $\lambda = 0.01$	0.987 (0.011) $\lambda = 0.0001$	0.977 (0.001) $\lambda = 0.0001$	0.754 (0.003) $\lambda = 0.01$	1.088 (0.006) $\lambda = 0.0001$	0.797 (0.002) $\lambda = 0.01$	0.509 (0.006) $\lambda = 0.1$	0
LOS-ℓ_2	0.704 (0.002) $\lambda = 0.01$	0.723 (0.006) $\lambda = 0.01$	0.716 (0.003) $\lambda = 0.01$	0.935 (0.005) $\lambda = 0.1$	0.985 (0.012) $\lambda = 0.0001$	0.977 (0.001) $\lambda = 0.0001$	0.748 (0.004) $\lambda = 0.01$	1.086 (0.007) $\lambda = 0.0001$	0.790 (0.001) $\lambda = 0.1$	0.505 (0.004) $\lambda = 0.01$	0

MSE values in the table are the average (and standard deviation) of 100 randomized experimental trails; the MSE of each trial is taken over a 10-fold cross validation. Bold indicates the lowest MSE at a 5% statistical significance based on a two-sample t-test.

Thus, it combines the benefits of both linear regression as well as LOS regression. Experiments on real-world benchmark data sets indicated the superior performance of ELOS over linear and LOS regression. Furthermore, ELOS maintains the explainability of learned solutions since we can tease apart the treatment of each feature based on its sort position.

Future work will extend the application of ELOS to decision fusion problems. We will also explore the application of regularization strategies that force the learned weights of ELOS towards predefined structures, which will enable us to identify data sets on which a simpler

model like an LOS or a linear regression may be a better fit. We will also explore how ELOS can be instantiated in deep learning architectures, providing explainable layers for deep networks.

Chapter 3

Explainable Choquet Integral Ridge Regression and Visualization

3.1 Introduction

Regression approaches typically seek a function, $f(\cdot)$, that can transform or map an independent variable, $\mathbf{x}$, to a dependent variable, y, given a training set of d-dimensional independent variables, $\mathbf{x} \in \mathcal{R}^d$, and 1-dimensional dependent variables, $y \in \mathcal{R}$. The relationship between $\mathbf{x}$ and y is a parameterized model (or function), such that

$$Y \approx r(\mathbf{x}, \alpha),$$

where α is the set of learned parameters of the regression function r. A linear model is a common choice for this function, $r(\mathbf{x}, \alpha) = \mathbf{w}^T\mathbf{x} + \beta$, where $\mathbf{w} \in \mathcal{R}^d$ and β is the bias. Using basis functions, we can extend this linear model to learn non-linear relations, $r(\mathbf{x}, \alpha) = \mathbf{w}^T\phi(\mathbf{x}) + \beta$, where ϕ is a set of basis functions. Examples for basis functions include quadratic, polynomial, and radial basis functions. Basis functions typically project the input data $\mathbf{x}$ into a higher-dimensional space, which allows the regression function to learn more complex non-linear relations (in the native space). However, inclusion of basis functions often eliminates the interpretability of the results. This is because, without basis functions, the learned parameter vector $\mathbf{w}$ directly indicates how each variable in the input $\mathbf{x}$ affects the output, but when basis functions are included, the parameter vector $\mathbf{w}$ is not as interpretable.

The regression parameters α are typically trained using a set of training data pairs $(X, Y) = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. The training process optimizes the regression parameters α with respect to an error function, usually squared-error:

$$\alpha^* = \arg\min_{\alpha} \sum_{i=1}^{n} \left(r(\mathbf{x}_i, \alpha) - y_i\right)^2 .$$

This is the well-known least-squares problem. For more extensive details on regression, in general, we suggest [58].

In our recent previous work [59], we proposed a regression model based on the ChI with

respect to a BC [52], where we demonstrated that ChI, in combination with an FM can produce a non-linear aggregation method, which essentially is a compressed parameterization of a set of linear convex sums, one for each sort order of the input. Since the number of parameters in an FM scales as 2^d, where d is the number of variables in the input, training the ChI-FM regression model on a high-dimensional input data requires a large training data set with enough *rank-independent* observations. However, many real-world data sets often do not contain enough *rank-independent* observations; training on such data sets typically results in an overfit model that does not generalize well on the unseen data. In this chapter, we build on the prior work by addressing the overfitting problem of ChI-FM regression methods by the introduction of ℓ_2-regularization in the training process.

Several previous works have explored regression using ChIs [50, 77, 116]. Some works explored certain types of FMs, such as Sugeno's λ-measure, P-measures, or L-measures, e.g., [77]. Different from these works, the learning approach of our method enables us to learn *any* bounded capacity, which is more general than a parameterized FM. Grabisch [50] proposed a method to learn an FM using training data; however, since this approach is limited to FMs, it has limited applicability. In addition, this approach used a single bias value model (similar to our CIR(1) model proposed here), which limits the flexibility of the method. Our method learns a more flexible bias model (up to one bias for each possible sort). ChI was also applied to logistic regression by Tehrani et al. [116]; this approach also included a single bias value (like our CIR(1) model). Our method could be used to extend this logistic regression approach. Recently, Du and Zare [36] proposed a multiple instance

learning ChI regression, though a bias model was not included in this regression approach.

Our method [59] was a significant extension of these prior works as it used a bounded capacity and a bias model that allowed encoding of a linear model for each possible sort in the input data. In [59] our experimental results showed better performance than comparable methods. Work presented in this chapter further improves our method by applying ℓ_2-regularization. Experimental results show that the application of ℓ_2-regularization has consistently improved the performance as compared to the unregularized version.

ChIs with respect to non-monotonic measures have been discussed in [92, 94, 95, 123]. These works provide a good basis for the development and application of ChIs with respect to non-monotonic measures. Our work here is a significant extension to these works to enable a generalized ChI-based regression model along with regularization.

3.2 Choquet Integral Ridge Regression

An FM defined in 1.2.1, when used for aggregation with ChI, results in convex sums of input variables that are bounded between maximum and minimum values of the input variables. To enable a continuous unbounded aggregation of outputs, we will relax the properties of the FM to a BC.

Definition 1. A BC is a set-valued function $f : \Omega \rightarrow \mathcal{R}$, with the boundary property

$f(\emptyset) = 0$.[1]

The discrete (finite Ω) ChI is defined as

$$\int_C \mathbf{h} \circ g = C_g(\mathbf{h}) = \sum_{i=1}^{d} h_{\pi(i)} \left[g(\Pi_i) - g(\Pi_{i-1}) \right], \tag{3.1a}$$

$$= \gamma_\pi^T \mathbf{h}_\pi, \tag{3.1b}$$

where π is a permutation of X, such that $h_{\pi(1)} \geq h_{\pi(2)} \geq \ldots \geq h_{\pi(d)}$, $\Pi_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$, and $g(\Pi_0) = 0$ [53, 110]. In (3.1b), we have simply reformulated (3.1a) as the dot-product of the vectors γ_π and $\mathbf{h}_\pi$, where

$$\mathbf{h}_\pi = \left(h_{\pi(1)},\ h_{\pi(2)},\ \ldots,\ h_{\pi(d)} \right)^T, \tag{3.2}$$

$$\gamma_\pi = \left(g(\Pi_1),\ (g(\Pi_2) - g(\Pi_1)),\ \ldots,\ (1 - g(\Pi_{d-1})) \right)^T. \tag{3.3}$$

The key insight from (3.1b) is that the ChI with respect to the FM is essentially a collection of $d!$ linear-order statistics on h, one for each possible sort order of the evidence h.[2] The elements of γ_π are simply the weights of each evidence value, as represented by the gain in the FM up through the lattice. More detail on the properties of fuzzy ChIs and fuzzy integrals in general can be found in [46, 53, 110].

[1] With regard to Choquet integral regression, described in Section 3.2.1, this boundary condition on f is arbitrary. It can be shown that any constant bias applied to all values in f does not change the result of the regression. This boundary condition can help with optimization from a practical computing standpoint.

[2] It can be further specified to say that the ChI is a collection of $d!$ *ordered weighted averages* (OWA), as $\sum_i (\gamma_\pi)_i = 1$.

3.2.1 Choquet Integral Regression

3.2.1.1 CIR formulation

We can directly extend the ChI at (3.1) to CIR by integrating with respect to a BC f and adding a bias term,

$$\int_C \mathbf{h} \circ f = C_f(\mathbf{h}) = \beta_\pi + \sum_{i=1}^{d} h_{\pi(i)} \left[f(\Pi_i) - f(\Pi_{i-1}) \right], \tag{3.4a}$$

$$= \beta_\pi + \rho_\pi^T \mathbf{h}_\pi, \tag{3.4b}$$

where π and Π are defined as in (3.1), $\mathbf{h}_\pi$ is defined at (3.3), $\beta_\pi \in \mathcal{R}$ is a bias term[3], and

$$\rho_\pi = \Big((f(\Pi_1) - f(\emptyset)),\ (f(\Pi_2) - f(\Pi_1)),\ \ldots, \\ (f(X) - f(\Pi_{d-1})) \Big)^T. \tag{3.5}$$

In an FM g, $f(\emptyset)$ and $f(X)$ are assigned the static boundary values of 0 and 1, respectively. We remove these boundary constraints in (3.5) by including $f(\emptyset)$ and $f(X)$ in ρ_π; see γ_π at (3.3) for comparison.

How is the CIR formulation at (3.4) a regression? Consider the formulation of CIR at

[3]We choose to generalize the bias so that one could have up to $d!$ different bias terms, one for each sort; however, as we will explore in Section 3.2.1.2, this may be computationally intractable—$d!$ can grow to be a large number—and, hence, we will also develop solutions that use fewer bias terms.

(3.4b)—with β_π as the learned bias and ρ_π as the learned weight vector, this clearly is in the form of linear regression. In addtion, since we removed the boundary constraints, the values in ρ_π can take any value in the set of reals, $\mathcal{R}$—see (3.5). This makes the CIR a compressed parameterization of a set of linear convex sums. In other words, CIR is a set of linear regressions, one for each of the $d!$ possible sorts of h. The ChI, since it is an aggregation operator, produces outputs that are bounded by the interval $[\min\{\mathbf{h}\}, \max\{\mathbf{h}\}]$; while the CIR, by allowing the learned parameters in ρ_π to take any value in $\mathcal{R}$, enables the mapping of inputs to anywhere in the set of reals, $C_f(h) \in \mathcal{R}$, and is therefore a regression operator.

3.2.1.2 CIR learning

Given a set of training data pairs $\{(\mathbf{h}_1, y_1), \ldots, (\mathbf{h}_n, y_n)\}$, $y_i \in \mathcal{R}, \forall i$, we would like to learn a prediction function o such that $o(\mathbf{h}_i) = y_i$. This is a standard regression problem. In our prior work [5, 6], we explored the approaches to train the ChI as a prediction function, by minimizing the SSE between the true output and the prediction for a given set of training data, i.e.,

$$g^* = \arg\min_g \left\{ \sum_{i=1}^{n} \left(C_g(\mathbf{h}_i) - y_i \right)^2 \right\}. \tag{3.6}$$

It can be shown that the solution to (3.6) is the *quadratic program* (QP)

$$\min_{\mathbf{u}_g} \mathbf{u}_g^T D \mathbf{u}_g + \mathbf{t}^T \mathbf{u}_g, \quad C\mathbf{u}_g \leq \mathbf{0}, \quad (\mathbf{0}, 1)^T \leq \mathbf{u}_g \leq \mathbf{1}, \tag{3.7}$$

where $\mathbf{u}_g$ is the lexographically-ordered FM g, i.e., $\mathbf{u}_g = (g(\{x_1\}), g(\{x_2\}), \ldots, g(\{x_1, x_2\}), g(\{x_1, x_3\}), \ldots, \quad g(\{x_1, x_2, \ldots, x_d\}))$; the matrices D and $\mathbf{t}$ are composed of the training data $\mathbf{h}$ and $\mathbf{y}$; and the matrix C enforces the monotonicity property on the learned FM g. Our prior work [6] contains the details on the construction of these matrices and the implementation of the QP. Using this QP-based learning approach, we applied ChI on many sensor fusion problems; however, the output of ChI is limited to the interval between the maximum and the minimum of the inputs. Hence, we explore next, the process to learn the CIR, which does not have such limitations.

This chapter focuses on the BC f and the CIR at (3.4). Therefore, we would like to solve the minimization problem

$$(f^*, \beta^*) = \arg\min_{f,\beta} \left\{ \sum_{i=1}^{n} (C_f(\mathbf{h}_i) - y_i)^2 \right\}. \tag{3.8}$$

First, we will rewrite (3.4) as

$$C_f(\mathbf{h}_i) = \beta_\pi + \sum_{j=0}^{d} f(\Pi_j) \left[(\mathbf{h}_i)_{\pi(j)} - (\mathbf{h}_i)_{\pi(j+1)} \right], \tag{3.9}$$

where we define $(\mathbf{h}_i)_{\pi(0)} = (\mathbf{h}_i)_{\pi(d+1)} = 0$, $\forall i$. Since $f(\Pi_0) = f(\emptyset) = 0$, the first summation term equals 0. However, if the user wants to set different boundary conditions, or none at all, this term can be adjusted accordingly. To continue, f is then lexographically ordered, i.e., $\mathbf{u}_f = (f(\emptyset), f(\{x_1\}), f(\{x_2\}), \ldots,$

$f(\{x_1, x_2\}), f(\{x_1, x_3\}), \ldots, f(\{x_1, x_2, \ldots, x_d\}))$.[4] The SSE term (3.8) on expansion gives

$$E^2 = \sum_{i=1}^{n} (C_f(\mathbf{h}_i) - y_i)^2 = \sum_{i=1}^{n} (H_i^T \mathbf{u}_f + B_i^T \beta - y_i)^2, \tag{3.10}$$

where H_i is a $2^d \times 1$ vector that contains the difference terms in (3.9); B_i is a $b \times 1$ bit vector that chooses (or computes) the bias from the vector β for a given training data pair (more on that soon).

To transform (3.10) into a standard least-squares problem, we concatenate a variable vector $\mathbf{u} = (\mathbf{u}_f, \beta)^T$, and then build the vector

$$D_i = \left(H_i^T, B_i^T\right). \tag{3.11}$$

Thus, (3.10) becomes

$$E^2 = \sum_{i=1}^{n} (D_i \mathbf{u} - y_i)^2, \tag{3.12}$$

[4] One could choose any ordering of f and derive the QP matrices appropriately. For example, in our code library we use binary ordering, such that $\mathbf{u}_f = (f(\emptyset), f(\{x_1\}), f(\{x_2\}), f(\{x_1, x_2\}), f(\{x_3\}), f(\{x_1, x_3\}), \ldots, f(X))$.

This has the standard least-squares solution of

$$\mathbf{u} = (D^T D)^{-1} D^T \mathbf{y}, \tag{3.13a}$$

$$D = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{bmatrix}. \tag{3.13b}$$

This solution is satisfying as it further bolsters our claim that CIR is regression at its core. We used the least-squares solver in Matlab for the results presented in this chapter. Problems such as singular matrices and underdetermined systems are automatically addressed by the solver. Note that the least-squares solution at (3.13) is a mathematically correct solution, but can be problematic due to its inverse; hence, least-square solvers are often more stable in practice.

We now describe the process to build the matrices H_i and B_i. The approach for a simple 3-input case is show in Example 2. It may be useful to follow along with this example as we describe the process in more detail.

The $2^d \times 1$ H_i matrix is designed to contain the $(d + 1)$ difference terms,

$\left[(\mathbf{h}_i)_{\pi(j)} - (\mathbf{h}_i)_{\pi(j+1)}\right]$, in (3.9). The rest of the elements of H_i are all zero. That is,

$$H_i = \begin{pmatrix} 0 - (\mathbf{h}_i)_{\pi(1)} \\ \vdots \\ 0 \\ \vdots \\ (\mathbf{h}_i)_{\pi(1)} - (\mathbf{h}_i)_{\pi(2)} \\ \vdots \\ 0 \\ \vdots \\ (\mathbf{h}_i)_{\pi(d)} - 0 \end{pmatrix}. \tag{3.14}$$

The B_i matrix varies based on the user's choice of the β vector construction. Though one could imagine numerous ways to learn the bias vector β, in this chapter, we present three possible choices:

1. Use a single scalar β value; thus, $B_i = 1, \forall i$. This is simplest and least computationally expensive choice.

2. Pick the β value based on the first element in the sort order $\pi(1)$, i.e., β is only dependent on the input with greatest magnitude. Thus, B_i is $d \times 1$ and takes the form $[B_i]_{\pi(1)} = 1$, else $[B_i] = 0$.

Table 3.1
Three Methods for Building Bias Vector β

Name	Description	B_i
1-bias	One bias term	$B_i = 1, \forall i$
d-bias	One term for each max-value in sort	$[B_i]_{\pi(1)} = 1$ else $[B_i] = 0$
2^d-bias	Computed bias for each possible sort	See (3.15)

3. The third method mimics the lattice of the BC and sets B_i according to the non-zero elements of H_i. That is,

$$[B_i]_j = \begin{cases} 1 & [H_i]_j > 0, \\ 0 & \text{else.} \end{cases} \tag{3.15}$$

In the third method, B_i is a $2^d \times 1$ matrix with $d + 1$ entries set to 1. Thus, for each sort order, the bias value in the regression solution is the sum of $d + 1$ elements of the β vector. In this way, the CIR can thus learn the β vector to produce a different bias for each sort order, but encoding this bias with only 2^d values (rather than $d!$). Table 4.1 outlines the three methods for learning the bias.

One could also imagine choosing a β value for each possible sort order; thus, B_i is $d! \times 1$ and has one entry that is set to 1 depending on the coding of the sort order. We view this choice as intractable in practice, as $d!$ can become very large, e.g., $10! = 3,628,800$.

Example 2. For this example, we will use a β vector that is $2^d \times 1$. Consider two training

data pairs ($n = 2$) with three inputs ($d = 3$) as follows:

$$(\mathbf{h}_1, y_1) = ((1, 2, 4)^T, 8),$$

$$(\mathbf{h}_2, y_2) = ((4, -6, 1)^T, 2).$$

The sort order for data pair 1 is $\Pi = (3, 2, 1)$, and the sort order for data pair 2 is $\Pi = (1, 3, 2)$. Thus, the QP sub-matrices for this example are

$$H_1 = (-4, 0, 0, 2, 0, 0, 1, 1)^T,$$

$$H_2 = (-4, 3, 0, 0, 0, 7, 0, -6)^T,$$

$$B_1 = (1, 0, 0, 1, 0, 0, 1, 1)^T,$$

$$B_2 = (1, 1, 0, 0, 0, 1, 0, 1)^T.$$

For this example, the least-squares problem is underdetermined. Using Matlab's solver, the solution is $\mathbf{u}_f = (-2, 0, 0, 0, 0, -0.86, 0, 0)^T$, and $\beta = \mathbf{0}$. We now extend CIR with regularization to account for overfitting in learning.

3.2.2 CIR with Ridge Regularization

Recall the insight of (3.4) (or (3.1b)) where we observe that the ChI encodes $d!$ regression models (or linear-order statistics), one for each of the $d!$ possible sort orders. Let

us enumerate these sorting orders as $\pi_1, \pi_2, ..., \pi_{d!}$, leading to the $d!$ regression models, $\rho_{\pi_1}, \rho_{\pi_2}, ..., \rho_{\pi_{d!}}$. We then modify the SSE cost function in (3.12) to include ℓ_2 (ridge) regularization terms for all $d!$ models, yielding

$$E_R^2 = \sum_{i=1}^{n} (D_i \mathbf{u} - y_i)^2 + \lambda \sum_{j=1}^{d!} \|\rho_{\pi_j}\|_2^2, \tag{3.16}$$

where λ is the regularization parameter defining the weight of the regularizer—a user-tuned quantity. To solve this, we must express the various regression models, ρ_{π_j}, in terms of the BC, $\mathbf{u}_f$. This is accomplished by defining a $d \times 2^d$ matrix $A_{\rho_{\pi_i}}$ to sift the regression model from $\mathbf{u}_f$, i.e.,

$$\rho_{\pi_i} = A_{\rho_{\pi_i}} \mathbf{u}_f. \tag{3.17}$$

Note the matrix $A_{\rho_{\pi_i}}$ is mostly zeros, and each row has at-most two non-zero elements (one $+1$ and one -1); this matrix-vector product produces the difference terms shown in (3.5). Also note that since the ridge regression term at (3.16) is applied to the product $A_{\rho_{\pi_i}} \mathbf{u}$, we can interpret the term as a Tikhonov regularizer and the matrix $A_{\rho_{\pi_i}}$ can be thought of as a Tikhonov matrix [121].

Each sort order will have its own $A_{\rho_{\pi_i}}$, thus there will be $d!$ unique sifting matrices. We

now express (3.16) in terms of $\mathbf{u}_f$ as

$$E_R^2 = \sum_{i=1}^{n}(D_i\mathbf{u} - y_i)^2 + \lambda \sum_{j=1}^{d!} \rho_{\pi_j}^T \rho_{\pi_j} \tag{3.18a}$$

$$= \sum_{i=1}^{n}(D_i\mathbf{u} - y_i)^2 + \lambda \sum_{j=1}^{d!} \mathbf{u}_f^T A_{\rho_{\pi_j}}^T A_{\rho_{\pi_j}} \mathbf{u}_f \tag{3.18b}$$

$$= \sum_{i=1}^{n}(D_i\mathbf{u} - y_i)^2 + \lambda \mathbf{u}_f^T \hat{G}_\rho \mathbf{u}_f, \tag{3.18c}$$

where $\hat{G}_\rho = \sum_{j=1}^{d!} \left(A_{\rho_{\pi_j}}^T A_{\rho_{\pi_j}} \right)$. Note that $\hat{G}_\rho$ is a constant sparse matrix of size $2^d \times 2^d$ and only depends on d, thus it can be built offline. Finally, appending block matrices of zeros to $\hat{G}_\rho$ to make it compatible with the concatenated vector $\mathbf{u}$ allows the ridge cost function to be written as

$$E_R^2 = \sum_{i=1}^{n}(D_i\mathbf{u} - y_i)^2 + \lambda \mathbf{u}^T G_\rho \mathbf{u}, \tag{3.19}$$

where $G_\rho = \begin{pmatrix} \hat{G}_\rho & 0 \\ 0 & 0 \end{pmatrix}$. Unfolding the squared term in (3.19), setting the gradient with respect to $\mathbf{u}$ equal to zero, and solving for $\mathbf{u}$ gives the minimizer

$$\hat{\mathbf{u}}_R = \left(D^T D + \lambda G_\rho \right)^{-1} D^T \mathbf{y}. \tag{3.20}$$

This solution is satisfying since its form is very similar to the well-known ridge regression solution.[5] Note that if $G_\rho = I$, where I is the identity matrix, the solution matches that of

[5]The ridge regression minimizer, when learning regression weight vectors directly is $\hat{\mathbf{w}}_R =$

ridge regression exactly. This is not surprising; because the fuzzy measure vector we are learning here has a different structure than the typical regression weight vector, the regularization matrix, G_ρ, must have a different structure than the identity matrix to compensate.

It is also interesting to note that since ρ_{π_i} is defined by a subset of $\mathbf{u}_f$, the objective function at (3.16) is essentially group lasso performed on $\mathbf{u}_f$ [136]. Furthermore, since each individual element of $\mathbf{u}_f$ appears in more than one ρ_{π_i}, it is the more general case of group lasso with overlapping groups [67, 135].

3.2.3 Experiments

We evaluated the performance of our CIR methods and the impact of ℓ_2-regularization using real world data sets from the UCI Machine Learning repository [33]. In addition, we compared the performance with several other regression methods; Table 3.2 presents the details on methods used in our experiments. The methods *Interactions*, *PureQuadratic*, and *Quadratic* use basis functions (indicated by $\phi(\mathbf{h})$ in Table 3.2) to project the input data into a higher-dimensional space and thereby induce non-linearity into the regression solutions. Note that these are not the only multivariate regression methods that exist; one could choose from a whole host of basis functions, non-parametric predictors, etc. We chose these methods since we consider them to be a fair comparison of "simple" regression

$\left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$, where X is a data (or design) matrix, $\mathbf{y}$ is a vector of target outputs, and I is the identity matrix [58].

Table 3.2
Regression Methods

Name	Model	Model Dim.	Comments
Linear	$y = \mathbf{w}^T\mathbf{h} + \beta$	$d+1$	Standard method
Interact	$y = \mathbf{w}^T\phi(\mathbf{h}) + \beta$	d^2+1	Linear plus pair-products (no square terms)
PureQuad	$y = \mathbf{w}^T\phi(\mathbf{h}) + \beta$	$2d+1$	Linear plus square terms (no *Interact* terms)
Quad	$y = \mathbf{w}^T\phi(\mathbf{h}) + \beta$	d^2+2d+1	Linear plus *Interact* and *PureQuad* terms
LOS	$y = \mathbf{w}^T\mathbf{h}_\pi + \beta$	$d+1$	Sorts input first
CIR(b)	$y = C_f(\mathbf{h})$	$2^d + b$	b indicates bias model $\{1, d, 2^d\}$

methods. We implemented these methods using the `fitlm` function in Matlab with corresponding model specification: `interactions`, `purequadratic`, and `quadratic`, respectively. To solve *linear order statistic* (LOS) regression [130], we first sort the inputs and then apply a linear regression. The sorting process induces the non-linearity of LOS. The CIR β model that is used is indicated by CIR(b) where b indicates the number of bias terms $\{1, d, 2^d\}$—see Table 4.1.

3.2.4 Impact of ridge regression on CIR methods

Using 10 real-world data sets from the UCI machine learning repository [33], we compared the performance of CIR methods with and without the application of ridge regression. Table 3.3 shows the performance of the CIR(b) methods[6] and their corresponding ridge-regularized methods. The MSE values presented in the table are the average values taken over 100 experimental trails—MSE of each trial is taken over a 10-fold cross validation. Based on a two-sample t-test, we identified the instances where the ridge regularization has improved the performance at a 5% statistical significance level. Overall, the application of ridge regression has improved the performance of CIR(1) in six out of ten instances, and for both the CIR(d) and CIR(2^d) methods, we observed improved performance in eight out of ten instances.

[6]Since the number of parameters in the BC for CIR methods scales as 2^d where d is the number of features, computation becomes intractable for larger values of d, e.g., 10!=3,628,800. Therefore, for the data sets with more than six features, we applied *principal component analysis* (PCA) to reduce the dimensionality to six features, and then applied the CIR methods.

Table 3.3
Impact of ridge regression on CIR methods*

Method	Concrete	Real Estate	Fish Toxicity	Aquatic Toxicity	Red Wine	White Wine	ENB-2	Yacht	Airfoil	ISE
n	1,030	414	908	546	1599	4898	768	308	1503	536
d	8	5	6	8	11	11	8	6	5	7
CIR(1)	0.315 (0.003)	0.364 (0.008)	0.421 (0.007)	0.629 (0.333)	0.658 (0.004)	0.729 (0.001)	0.056 (0.001)	0.157 (0.041)	0.338 (0.002)	0.534 (0.012)
CIR(1)-Ridge	**0.311 (0.003)** $\lambda = 0.01$	**0.345 (0.004)** $\lambda = 0.1$	**0.410 (0.004)** $\lambda = 0.1$	0.513 (0.004) $\lambda = 0.1$	**0.648 (0.004)** $\lambda = 0.1$	**0.726 (0.001)** $\lambda = 0.1$	0.055 (0.001) $\lambda = 0.001$	0.150 (0.006) $\lambda = 0.0001$	0.338 (0.002) $\lambda = 0.001$	**0.479 (0.006)** $\lambda = 0.1$
CIR(d)	0.311 (0.004)	0.365 (0.007)	0.423 (0.015)	0.618 (0.122)	0.655 (0.004)	0.725 (0.002)	0.052 (0.001)	0.151 (0.015)	0.336 (0.001)	0.542 (0.015)
CIR(d)-Ridge	**0.307 (0.003)** $\lambda = 0.01$	**0.347 (0.004)** $\lambda = 0.1$	**0.410 (0.003)** $\lambda = 0.1$	**0.515 (0.005)** $\lambda = 0.1$	**0.645 (0.003)** $\lambda = 0.1$	**0.723 (0.001)** $\lambda = 0.1$	**0.055 (0.001)** $\lambda = 0.0001$	0.145 (0.008) $\lambda = 0.0001$	0.336 (0.002) $\lambda = 0.0001$	**0.487 (0.005)** $\lambda = 0.1$
CIR(2^d)	0.302 (0.005)	0.358 (0.011)	0.462 (0.011)	0.800 (0.101)	0.683 (0.007)	0.724 (0.002)	0.222 (0.412)	0.189 (0.031)	0.315 (0.002)	0.630 (0.021)
CIR(2^d)-Ridge	**0.294 (0.004)** $\lambda = 0.01$	**0.341 (0.007)** $\lambda = 0.1$	**0.436 (0.005)** $\lambda = 0.1$	**0.546 (0.010)** $\lambda = 0.1$	**0.670 (0.005)** $\lambda = 0.1$	**0.721 (0.002)** $\lambda = 0.1$	0.053 (0.001) $\lambda = 0.001$	**0.152 (0.018)** $\lambda = 0.001$	0.315 (0.002) $\lambda = 0.001$	**0.552 (0.010)** $\lambda = 0.1$

*MSE values in the table are the average of 100 experimental trails—MSE of each trial is taken over a 10-fold cross validation.
Bold indicates that the ℓ_2-regularization has improved the performance of that CIR method at a 5% statistical significance.

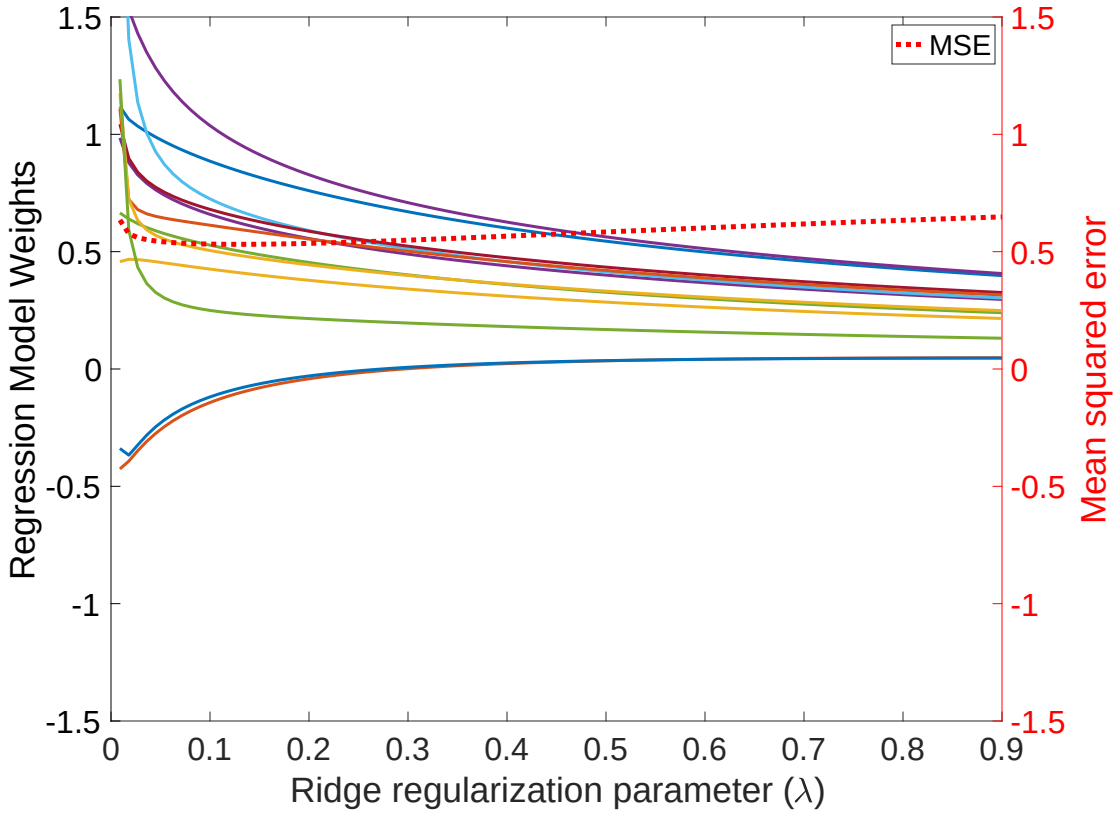

Figure 3.1: Impact of ridge regularization on the CIR(1) BC parameters learned on Aquatic Toxicity data set.

We increased the ridge regularization parameter (λ) in small steps and observed the MSE as well as the shrinkage of the learned regression parameters (i.e., the regression models sifted from the BC via (3.17)) for the CIR methods. The shrinkage plots in Figs. 3.1 and 3.2 demonstrate how the regularization restricted the magnitude of the parameters[7] with the increasing λ values. In both the examples, the best MSE was observed between the λ values of 0.01 and 0.1.

[7]The data sets Aquatic Toxicity and Instanbul Stock Exchange have eight features and seven features respectively. However, since we applied PCA to reduce the dimensionality to six features, the learned BCs for both the data sets have 2^6=64 parameters. In Figs. 3.1 and 3.2, for demonstration purpose, we are showing the trend of a randomly selected 12 out of the 64 BC parameters.

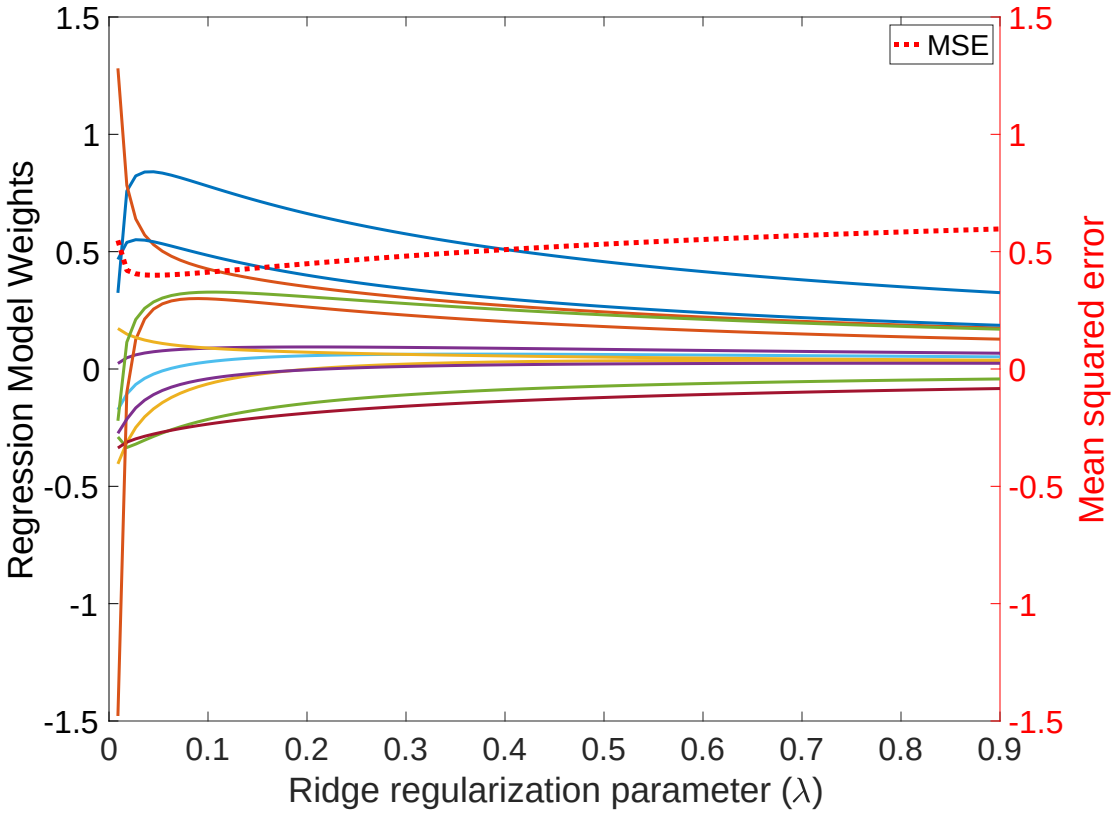

Figure 3.2: Impact of ridge regularization on the CIR(1) BC parameters learned on Istanbul Stock Exchange data set.

3.2.5 Performance of CIR methods

Table 4.2 shows the regression results of the CIR methods and the comparison algorithms for several real-world data sets. For each data set, the table shows the number of objects n and the number of features d. The best algorithms for each data set are shown in bold font. We performed a two-sample t-test at a 5% significance level to determine the statistically best results; hence, more than one algorithm can be considered as best. Overall, CIR methods produced best results on six out of 10 data sets, and CIR(1) with ridge regression

Table 3.4
MSE on Benchmark Data Sets*

Method	Concrete	Real Estate	Fish Toxicity	Aquatic Toxicity	Red Wine	White Wine	ENB-2	Yacht	Airfoil	ISE	# of best instances
n	1,030	414	908	546	1599	4898	768	308	1503	536	-
d	8	5	6	8	11	11	8	6	5	7	-
Linear	0.424 (0.002)	0.445 (0.002)	0.437 (0.002)	0.553 (0.003)	0.661 (0.001)	0.729 (0.001)	0.087 (0.000)	0.520 (0.003)	0.505 (0.001)	**0.437 (0.003)**	1
Linear-ℓ_1	0.418 (0.003) $\lambda = 0.01$	0.445 (0.002) $\lambda = 0.01$	0.435 (0.002) $\lambda = 0.01$	0.550 (0.005) $\lambda = 0.01$	0.660 (0.001) $\lambda = 0.01$	0.728 (0.001) $\lambda = 0.01$	0.087 (0.001) $\lambda = 0.0001$	0.519 (0.004) $\lambda = 0.0001$	0.501 (0.001) $\lambda = 0.01$	**0.437 (0.003)** $\lambda = 0.001$	1
Linear-ℓ_2	0.411 (0.002) $\lambda = 0.1$	0.445 (0.002) $\lambda = 0.01$	0.434 (0.002) $\lambda = 0.1$	0.552 (0.004) $\lambda = 0.1$	0.658 (0.001) $\lambda = 0.1$	0.728 (0.001) $\lambda = 0.1$	0.087 (0.001) $\lambda = 0.0001$	0.519 (0.004) $\lambda = 0.0001$	0.493 (0.001) $\lambda = 0.1$	**0.437 (0.003)** $\lambda = 0.001$	1
Interactions	0.358 (0.003)	0.364 (0.004)	0.417 (0.004)	0.764 (0.105)	0.658 (0.004)	0.763 (0.004)	0.053 (0.000)	0.409 (0.009)	0.371 (0.001)	0.485 (0.009)	0
Quadratic	**0.229 (0.001)**	0.349 (0.005)	0.424 (0.005)	0.714 (0.141)	0.657 (0.005)	0.774 (0.007)	**0.011 (0.000)**	0.087 (0.002)	0.367 (0.002)	0.509 (0.010)	2
Pure Quadratic	0.264 (0.001)	0.367 (0.004)	0.428 (0.003)	0.552 (0.007)	0.665 (0.003)	0.758 (0.002)	0.078 (0.000)	**0.080 (0.001)**	0.454 (0.001)	0.462 (0.006)	1
LOS	0.708 (0.002)	0.727 (0.005)	0.718 (0.003)	0.940 (0.007)	0.996 (0.002)	0.977 (0.001)	0.758 (0.003)	1.086 (0.009)	0.797 (0.001)	0.508 (0.008)	0
LOS-ℓ_1	0.707 (0.002) $\lambda = 0.01$	0.727 (0.007) $\lambda = 0.001$	0.709 (0.002) $\lambda = 0.01$	0.934 (0.007) $\lambda = 0.01$	0.987 (0.011) $\lambda = 0.0001$	0.977 (0.001) $\lambda = 0.0001$	0.754 (0.003) $\lambda = 0.01$	1.088 (0.006) $\lambda = 0.0001$	0.797 (0.002) $\lambda = 0.01$	0.509 (0.006) $\lambda = 0.1$	0
LOS-ℓ_2	0.704 (0.002) $\lambda = 0.01$	0.723 (0.006) $\lambda = 0.01$	0.716 (0.003) $\lambda = 0.01$	0.935 (0.005) $\lambda = 0.1$	0.985 (0.012) $\lambda = 0.0001$	0.977 (0.001) $\lambda = 0.0001$	0.748 (0.004) $\lambda = 0.01$	1.086 (0.007) $\lambda = 0.0001$	0.790 (0.001) $\lambda = 0.1$	0.505 (0.004) $\lambda = 0.01$	0
CIR(1)	0.315 (0.003)	0.364 (0.008)	0.421 (0.007)	**0.629 (0.333)**	0.658 (0.004)	0.729 (0.001)	0.056 (0.001)	0.157 (0.041)	0.338 (0.002)	0.534 (0.012)	1
CIR(d)	0.311 (0.004)	0.365 (0.007)	0.423 (0.015)	0.618 (0.122)	0.655 (0.004)	0.725 (0.002)	0.052 (0.001)	0.151 (0.015)	0.336 (0.001)	0.542 (0.015)	0
CIR(2^d)	0.302 (0.005)	0.358 (0.011)	0.462 (0.011)	0.800 (0.101)	0.683 (0.007)	0.724 (0.002)	0.222 (0.412)	0.189 (0.031)	**0.315 (0.002)**	0.630 (0.021)	1
CIR(1)-Ridge	0.311 (0.003) $\lambda = 0.01$	**0.345 (0.004)** $\lambda = 0.1$	**0.410 (0.004)** $\lambda = 0.1$	**0.513 (0.004)** $\lambda = 0.1$	**0.648 (0.004)** $\lambda = 0.1$	0.726 (0.001) $\lambda = 0.1$	0.055 (0.001) $\lambda = 0.001$	0.150 (0.006) $\lambda = 0.0001$	0.338 (0.002) $\lambda = 0.001$	0.479 (0.006) $\lambda = 0.1$	4
CIR(d)-Ridge	0.307 (0.003) $\lambda = 0.01$	0.347 (0.004) $\lambda = 0.1$	**0.410 (0.003)** $\lambda = 0.1$	**0.515 (0.005)** $\lambda = 0.1$	**0.645 (0.003)** $\lambda = 0.1$	0.723 (0.001) $\lambda = 0.1$	0.055 (0.001) $\lambda = 0.0001$	0.145 (0.008) $\lambda = 0.0001$	0.336 (0.002) $\lambda = 0.0001$	0.487 (0.005) $\lambda = 0.1$	3
CIR(2^d)-Ridge	0.294 (0.004) $\lambda = 0.01$	**0.341 (0.007)** $\lambda = 0.1$	0.436 (0.005) $\lambda = 0.1$	0.546 (0.010) $\lambda = 0.1$	0.670 (0.005) $\lambda = 0.1$	**0.721 (0.002)** $\lambda = 0.1$	0.053 (0.001) $\lambda = 0.001$	0.152 (0.018) $\lambda = 0.001$	**0.315 (0.002)** $\lambda = 0.001$	0.552 (0.010) $\lambda = 0.1$	3

*MSE values in the table are the average of 100 experimental trails—MSE of each trial is taken over a 10-fold cross validation.
Bold indicates lowest MSE at a 5% statistical significance.

was the best algorithm with best results on four out of 10 data sets. Even for the cases where the CIR methods were not the best, the ridge regularized CIR methods produced good(-enough) results.

3.2.6 Conclusions and Future Work

We reviewed our previously developed method of learning regression models using the Choquet fuzzy integral [59] and enhanced this method by applying ℓ_2 regularization. Experimental results with real-world data sets demonstrated that the introduction of regularization significantly improved the performance of CIR methods in 22 out of 30 (73%) instances. We then showed that ℓ_2 regularized CIR methods were superior to competing regression models on real-world benchmark data.

Future work will include extending the CIR method to deep learning architectures. In a recently published work [65] we show that the Choquet integral can be built using standard neural network operations and can thus be used to represent learned layers in deep networks. We also will extend the *explainable AI* (XAI) approaches proposed in [93, 102], which allow interpretation of the result of the Choquet integral.

3.3 Visualization of the CIR

In this section, we extend our previously proposed method [100] for visualization of FMs to that of BCs. Using an example data set, we then discuss the methods to visualize the Shapley and interaction indices of a BC of a CIR model.

3.3.1 Visualization of the BC

A convenient method to visualize an FM—and, thus, a BC—is to represent it as a lattice (i.e., Hasse diagram); Fig. 3.3 shows the lattice of a BC, which also happens to be an FM, for the case of three sources ($d = 3$). The size of individual nodes (BC variables) in the BC lattice are scaled proportional to their values.

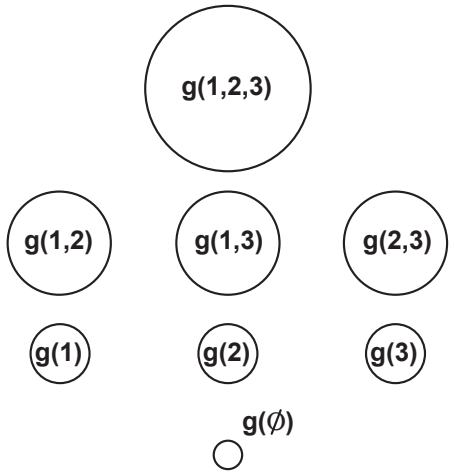

Figure 3.3: Lattice of BC elements for $d = 3$. Note that shorthand notation is used where $f(2, 3)$ is equivalent to $f(\{x_2, x_3\})$[100]

Fig. 3.5 is a visualization of the BC lattice learned by CIR for the regression of the Airfoil data[8]. Here, the black colored nodes are positive values, while the red nodes are negative. The thin white in-circles present in some of the nodes indicate the reduced magnitude of the node's value on application of ℓ_2 regularization.

[8]This is a NASA data set that comprises Airfoils of different sizes tested at various wind tunnel speeds and angles of attack. The target feature is the output sound pressure level measured in decibels. For more details on this regression problem, you can refer to [18]

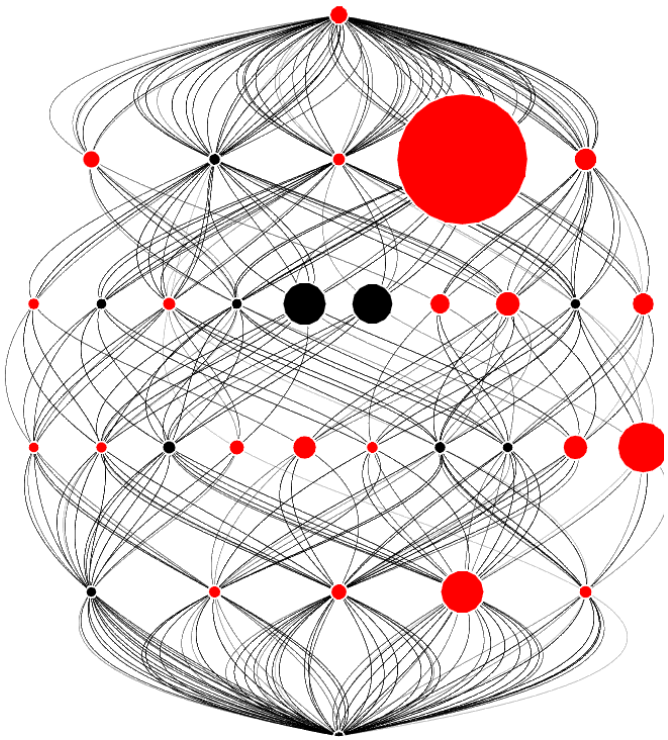

Figure 3.4: Lattice of learned BC and paths for the training data from the Airfoil data set [18]. The arrangement of the nodes follows that of Fig. 3.3, where each level in the lattice represents all subsets of equal cardinality. The black nodes in the lattice are positive values and the red nodes are negative.

3.3.2 Shapley and interaction indices

The ChI (and CIR) is parameterized by the capacity. Specifically, the capacity encodes the worth of all the individual subsets of sources and the ChI utilizes this information to aggregate the inputs (the integrand, h). It is important to note that the ChI operates on a weaker (and richer) premise than a great number of other aggregation operators that assume additivity (a stronger property than monotonicity). However, the capacity has a large

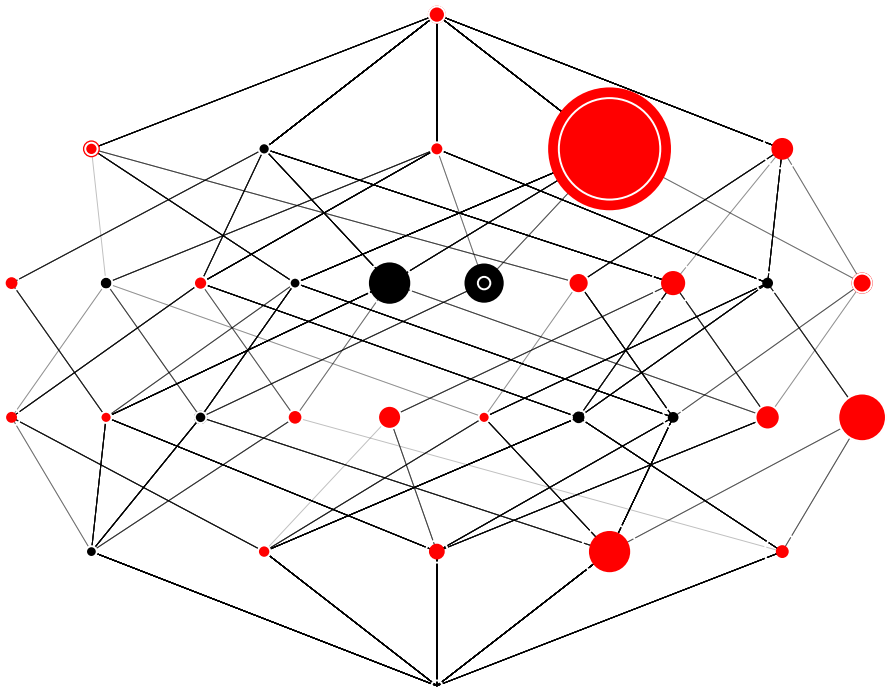

Figure 3.5: BC lattice of the CIR trained on Airfoil data set. Here, black nodes are positive values, while red are negative. The node sizes are scaled proportional to the magnitude of BC values. Thin white in-circles present in some of the nodes indicate the reduced magnitude of the node's value on application of ℓ_2 regularization—corresponding ℓ_2 regularization parameter value=0.001.

number of values; hence, it is not straightforward to interpret the ChI aggregation model, or to assess the relative significance of an individual source in the model. Answers to these complex questions are dispersed across the capacity; information theoretic indices aid us in summarizing such information from a capacity. The Shapley index has been proposed to summarize the worth of an individual source and the interaction index summarizes interactions between different sources.

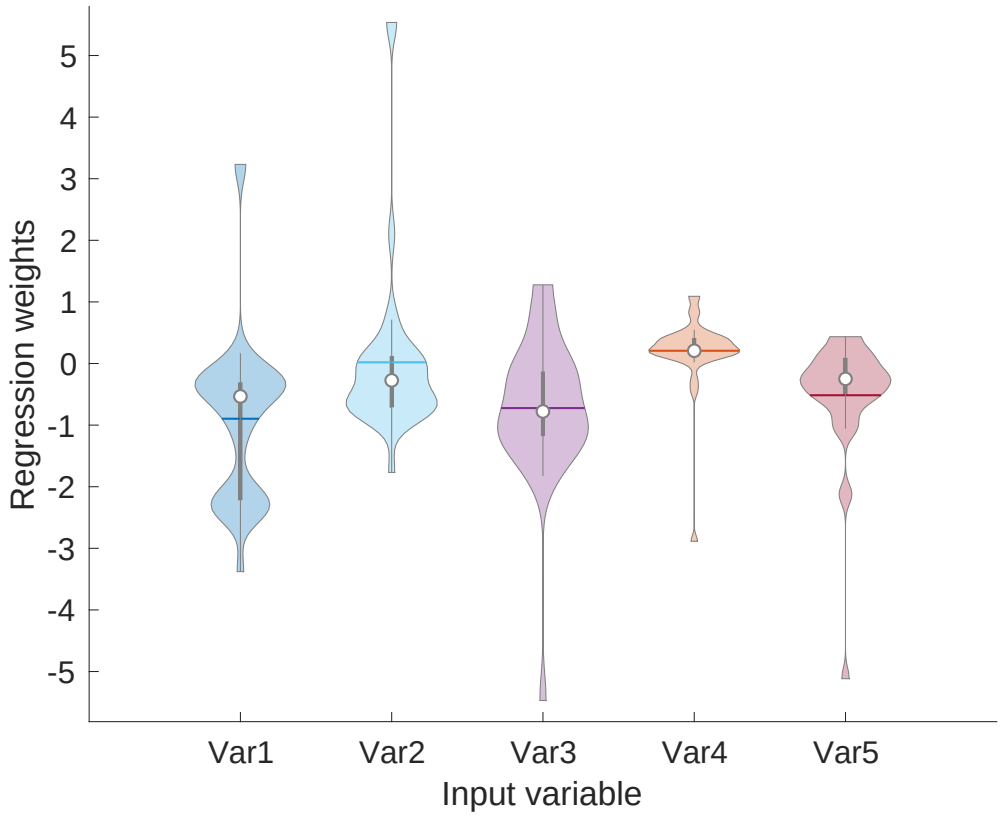

Figure 3.6: Violin plot of CIR weights for the Airfoil data set.

3.3.2.1 Shapley index

The Shapley values of an FM g are

$$\Phi_g(i) = \sum_{K \subseteq X \setminus \{i\}} \zeta_{X,1}(K)(g(K \cup \{i\}) - g(K)), \tag{3.21}$$

$$\zeta_{X,1}(K) = \frac{(|X| - |K| - 1)!|K|!}{|X|!} \tag{3.22}$$

where $K \subseteq X \setminus \{i\}$ denotes all proper subsets from X that do not include source i. The Shapley value of g is the vector $\Phi_g = (\Phi_g(1), \Phi_g(2), ..., \Phi_g(d))$ and $\sum_{i=1}^{d} \Phi(i) = 1$. The

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

We can adapt this to a BC f simply by substituting f for g in (3.21). In this case, the Shapley index represents the average weight of a source across all $d!$ possible sorts. The possible weights in the CIR for a given source i are the terms $f(K \cup \{i\}) - f(K)$, for all possible $K \subseteq X \backslash \{i\}$. It can be shown that there are 2^{d-1} possible values of $f(K \cup \{i\}) - f(K)$, which is simply the number of unique subsets in $X \backslash \{i\}$. However, each weight can appear in the CIR for different numbers of possible sorts—Example 3 illustrates this idea in detail.

Example 3. *Shapley index calculation* Consider three sources and focus on source 1. The weight that source 1 sees in the CIR calculation is determined by its sort value. Table 3.5 shows different weights assigned to source 1 depending on the sort order. We could notice that the same weight $(f(\{1\}) - f(\{\emptyset\}))$ was assigned to source 1 for the sorts 1,2,3 as well as 1,3,2. Similarly, the weight $(f(\{1,2,3\}) - f(\{2,3\}))$ was applied on source 1 for sorts 2,3,1 and 3,2,1. Thus, over $3! = 6$ sort orders, source 1 was assigned only four distinct CIR weights. Based on Table 3.5, we can calculate the Shapley index of source 1 as:

$$\begin{aligned}\frac{2}{6} \times \ (f(\{1\}) - f(\{\emptyset\})) + \frac{2}{6} \times (f(\{1,2,3\}) - f(\{2,3\})) \\ + \frac{1}{6} \times (f(\{1,2\}) - f(\{2\})) + \frac{1}{6} \times (f(\{1,3\}) - f(\{3\})).\end{aligned}$$

Table 3.5
Weight assigned to source 1 for different sort orders

Sort	CIR weight assigned to source 1
1,2,3 1,3,2	$(f(\{1\}) - f(\{\emptyset\}))$
2,3,1 3,2,1	$(f(\{1,2,3\}) - f(\{2,3\}))$
2,1,3	$(f(\{1,2\}) - f(\{2\}))$
3,1,2	$(f(\{1,3\}) - f(\{3\}))$

Example 3 shows how CIR assigns multiple sort-based weights for each of the three sources. We can assess how the feature variables affect the output by showing a violin plot of the frequency of each of the unique weight values attributed to each feature variable. Section 3.3 describes this visualization for a sample data set.

While the Shapley index—and the associated violin plot—is good for showing the contribution of each individual feature variable in the model, this index fails to show how feature variables interact. Hence, we now will describe the interaction index.

3.3.2.2 Interaction index

This index quantifies the complementary contributions of sets of sources [90]. While the interaction index can be generalized to any set of sources, we will focus on pairs of sources.

The interaction index between sources i and j is

$$I_g(i,j) = \sum_{K \subseteq X \setminus \{i,j\}} \zeta_{X,2}(K)(g(K \cup \{i,j\}) \\ - g(K \cup \{i\}) - g(K \cup \{j\}) + g(K)),\ i = 1,2,...,d,$$

$$\zeta_{X,2}(K) = \frac{(|X| - |K| - 2)!|K|!}{(|X| - 1)!} \tag{3.23}$$

Like the Shapley index, we can adapt the interaction index to the BC by directly substituting f for g. In this case, positive or negative values with larger relative magnitude imply a strong complementary contribution between sources i and j, where as values closer to zero would indicate redundancy. The reader can refer to Grabisch's work [56] for further details about the interaction index, its connections to game theory and interpretations. Next, we describe how to use the Shapley and interaction indices to produce visualizations that offer interpretation of the learned CIR model.

3.3.3 Visualization of Shapley index

The Shapley index and interaction indices provide good interpretability of the ChI with respect to an FM [98]. We can also use these to interpret the CIR model. The CIR is parameterized by the 2^d BC values for the aggregation of d features. Essentially, the CIR parameterizes a piece-wise linear regression model for each sort of the $d!$ possible sort

orders. Thus, a single trained BC enables learning a large number of weights for each feature, resulting in a (possibly) non-linear regression model. However, the model for each sort order still retains the interpretability of linear regression. The Shapley index at (3.21) provides a course interpretation of the average weight of each feature variable across all possible sorts. Thus, for a given feature variable, the Shapley index is a measure of the average regression weight of that feature across all possible sort orders. Hence, this index is analogous to the feature's coefficients in the linear regression models. While the Shapley index for a feature indicates the dominant relation between a feature and the target variable, we can analyze this relation at an even higher resolution by looking at the distribution of the contributing weight values for each feature. These weight values are $g(K \cup \{i\}) - g(K)$ for all sets $K \subseteq X \backslash \{i\}$.

Fig. 3.6 presents a violin plot of the weight values for each feature variable in the Airfoil data set [18]. The vertical spread of the weight values of a feature indicates the diversity in the sort order-based treatment of the variable. The average of these weight values is the Shapley index, marked by the horizontal lines in the plot, which provides a rough indication of the correlation of the features to the CIR output. For example, *Var4* is predominantly positively correlated with the output, while *Var3* is negatively correlated. The long vertical tails in the plot, e.g., seen in *Var2*, indicate the presence of a small number of observations (sort orders) where the general regression weights of the feature were significantly different from the Shapley index.

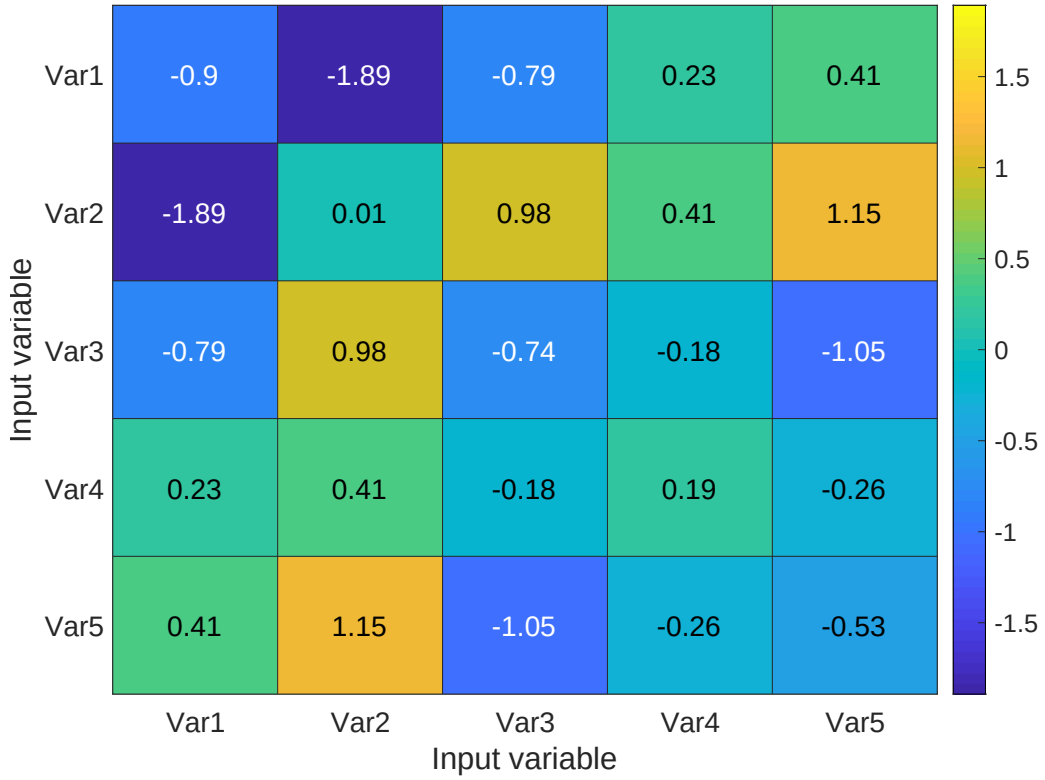

Figure 3.7: Heatmap of interaction indices for the Airfoil data set.

3.3.4 Visualization of Interaction index

The interaction index described in Section 3.3.2.2 can be visualized as an $d \times d$ matrix, where d is the number of input feature variables, and each cell (i, j) is the interact index for the variable pair (i, j). Fig. 3.7 presents a 5×5 heatmap of the interaction indices for the Aifoil data set. For a feature variable pair (i, j). An interaction index with relatively larger magnitude (postive/negative) represents a strong complementary contribution between i and j, where as values closer to 0 would indicate redundancy. The d values along the diagonal of these visualizations, i.e., interaction index of each input feature variable

with itself, are same as the Shapley indices, for those d input features, respectively. In this heatmap, we could notice that the Shapley index of *Var2* (cell $(2, 2)$) is 0.01—indicating a poor standalone impact on the regression output. However, *Var2*'s interaction indices with *Var1*, *Var3*, and *Var5* are -1.89, 0.98, and 1.15, respectively—indicating a strong complementary contribution of *Var2*. In this way, with the help of a heatmap visualization, we could disentangle the contribution and relevance of each variable individually, and while working in pairs with rest of the variables. Now, we apply these visualization methods on CIR models trained on real-world data sets, and explain the models.

3.3.5 Experiments

We trained CIR using real-world data sets from the UCI Machine Learning repository [33]. We evaluated the learned BCs using the Shapley and interaction indices to evaluate the significance of the contributing input feature variables. We visualized the BC lattices as Hasse diagrams and inferred the impact of ridge regularization. In Section 3.3, we introduced visualization methods using the regression model trained on the Airfoil data set. In this section, we train CIR on three more regression data sets and explain the learned models. Note that for each of these data sets, while training the CIR, we ran a grid search over the ℓ_2 regularization parameter λ—this section presents the best model (based on performance on the testing data) for each data set. Furthermore, to enable a balanced distribution of all possible sort orders, we z-normalized the data sets before splitting them into training and

testing portions, and corresponding target variables.

3.3.5.1 Fish Toxicity data set

This data set [21] was used to develop *quantitative structure–activity relationships* (QSAR) [21] models to predict acute aquatic toxicity towards the Pimephales Promelas (fathead minnow) fish. The data set comprises 6 real-valued features—the target feature is toxicity, measured as concentration of LC50. For more details on this regression problem, you can refer to [21].

Fig. 3.8 is a visualization of the learned BC. The diversity of node sizes and the distribution of black and red nodes is indicative of the variance in the sort order-based treatment of observations. This lattice has a healthy amount of such diversity; thus, CIR is a good choice for the regression of these data. It is also intuitively pleasing to see that the impact of ridge regularization was more pronounced on the larger nodes.

The spread of the Shapley values in Fig. 3.9 suggest that CIR is capturing a considerable amount additional non-linearity in the data compared to simpler methods like linear regression. Thus, CIR is a good choice for this problem.

From the heatmap of the interaction indices in Fig. 3.10, we could notice that feature variables *Var2* and *Var6* with Shapley values of 0.39 and 0.41, respectively, are the strongest

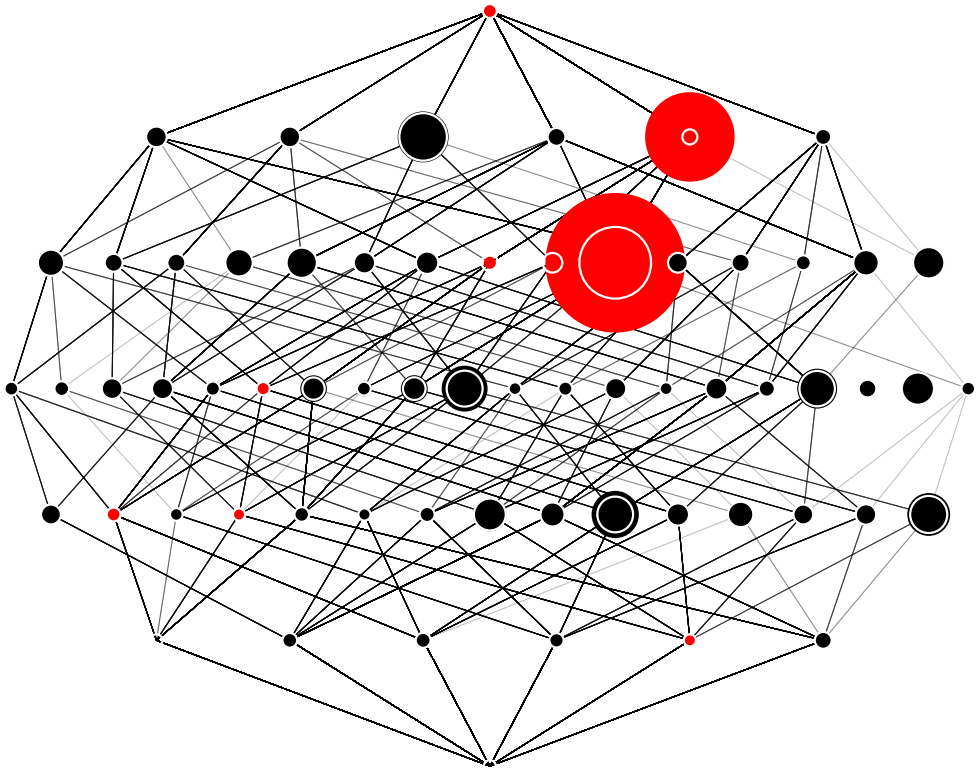

Figure 3.8: BC lattice of the CIR model trained on the Fish Toxicity data—ℓ_2 regularization parameter value= 0.001.

standalone contributors to the model. While *Var5*, with a shapley value of -0.04, though is a relatively poor individual contributor, its strong interaction indices with *Var3* and *Var6* indicate a sizable complimentary contribution to the regression output.

3.3.5.2 Real Estate data set

This is market historical data set of real estate valuation. The data is collected from Sindian Dist., New Taipei City, Taiwan. The regression problem is to predict the house price per

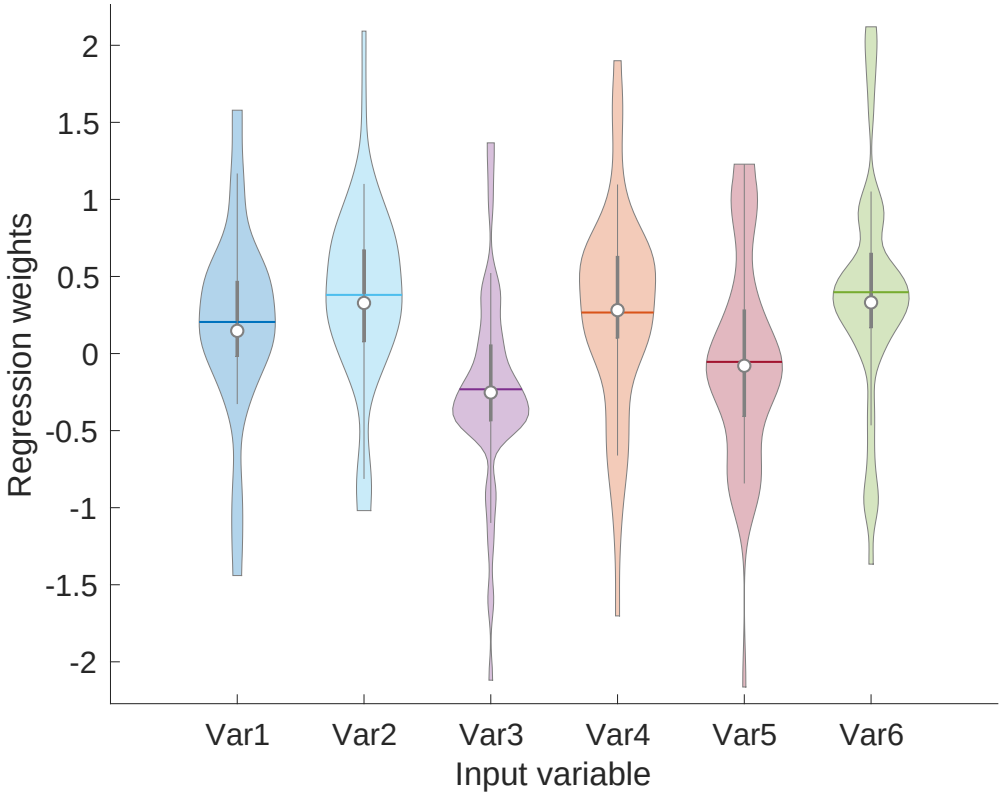

Figure 3.9: Violin plot CIR weights for the Fish Toxicity data.

unit area, using the real-valued features that include the age of the house, distance to the nearest MRT station, number of convenience stores nearby, and geographical coordinates. For more details on this problem, you can refer to [132].

The lattice visualization in Fig. 3.11 presents the BC trained for this CIR. While the node sizes in the lattice are relatively less diverse, we could notice a healthy share of both red and black nodes. On a portion of nodes, we could also notice the shrinkage of node values due to ℓ_2 regularization. The heatmap of the interaction indices in Fig. 3.13 signify that *Var2* is the strongest individual contributor to the model output, while the variables *Var3*, *Var4* and *Var5* have a considerable impact on the model output through the interactions

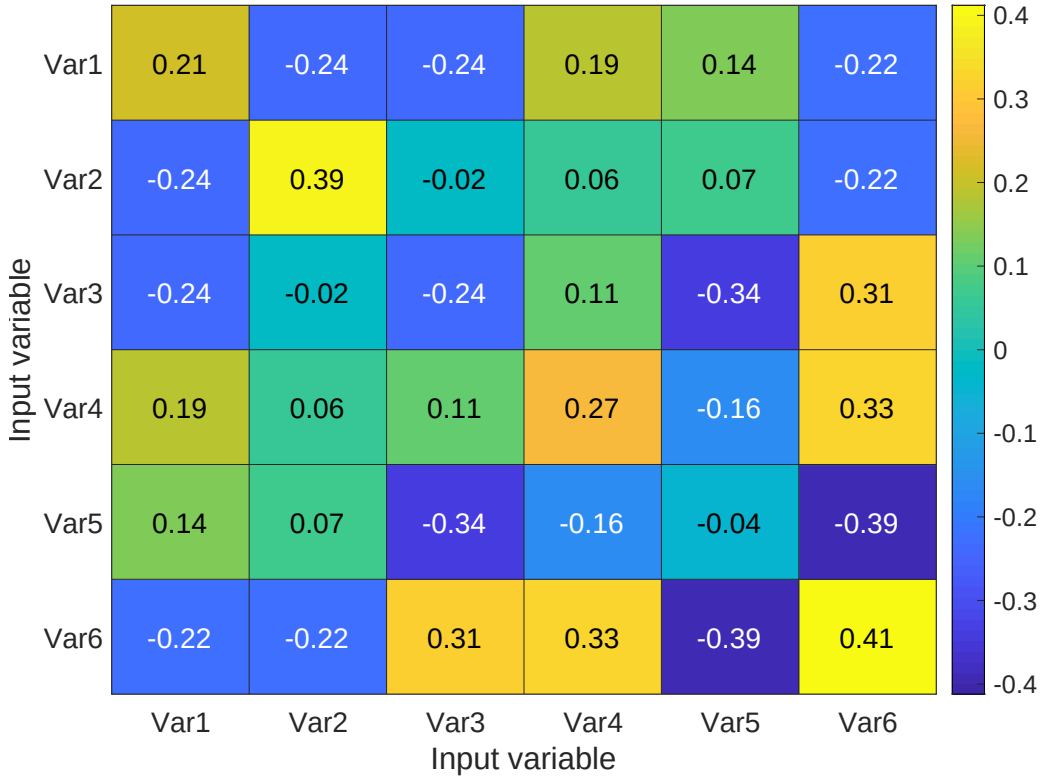

Figure 3.10: Heatmap of interaction indices for the Fish Toxicity data set.

among themselves. The violin plot of regression weights in Fig. 3.12 displays a healthy amount of vertical spread, indicating the diversity of the piece-wise linear regression models learned by the CIR. It is interesting to note that the violin plot of *Var2* has two distinct peaks—a simpler model like linear regression on this data would most likely learn the mean regression weight for *Var2*, marked by the horizontal line, and thus a sub-optimal solution.

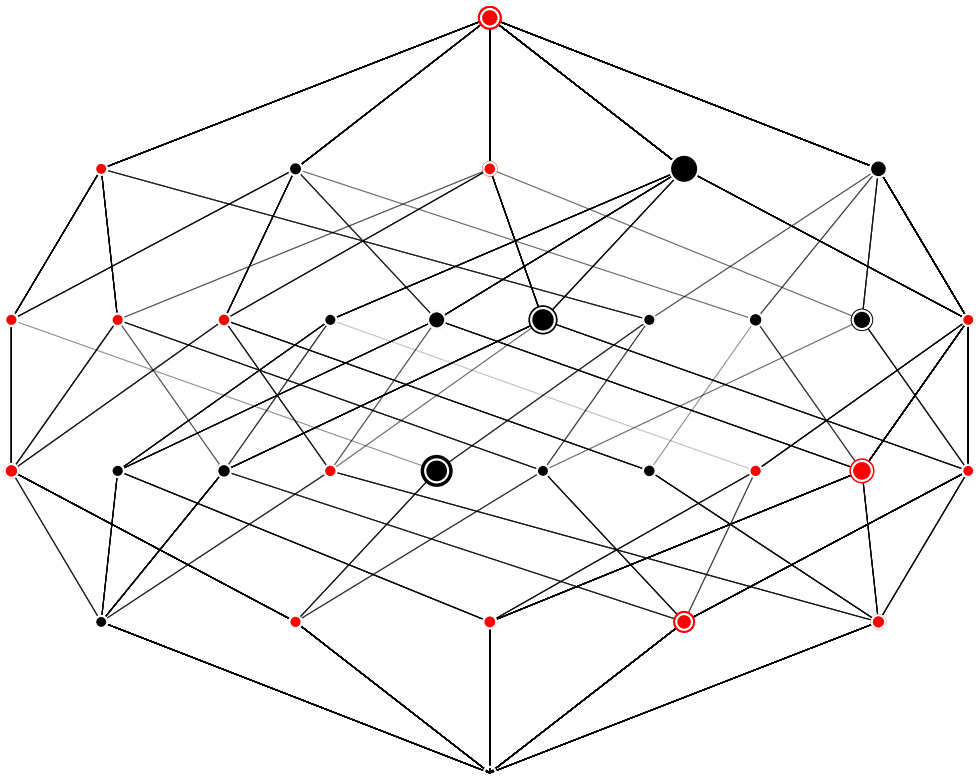

Figure 3.11: BC lattice of the CIR model trained on the Real Estate data—ℓ_2 regularization parameter value= 0.1.

3.3.5.3 Yacht data set

This data set comprises results from 308 experiments performed on 22 different hull forms. Using the basic hull dimensions and the boat velocity as inputs, the regression problem is to predict the residual resistance of sailing yachts. For more details on these experiments, you can refer to [43].

The BC lattice trained on Yacht data in Fig. 3.14 shows a good amount of diversity in both the size of nodes as well as the distribution of red and black nodes. However, since the

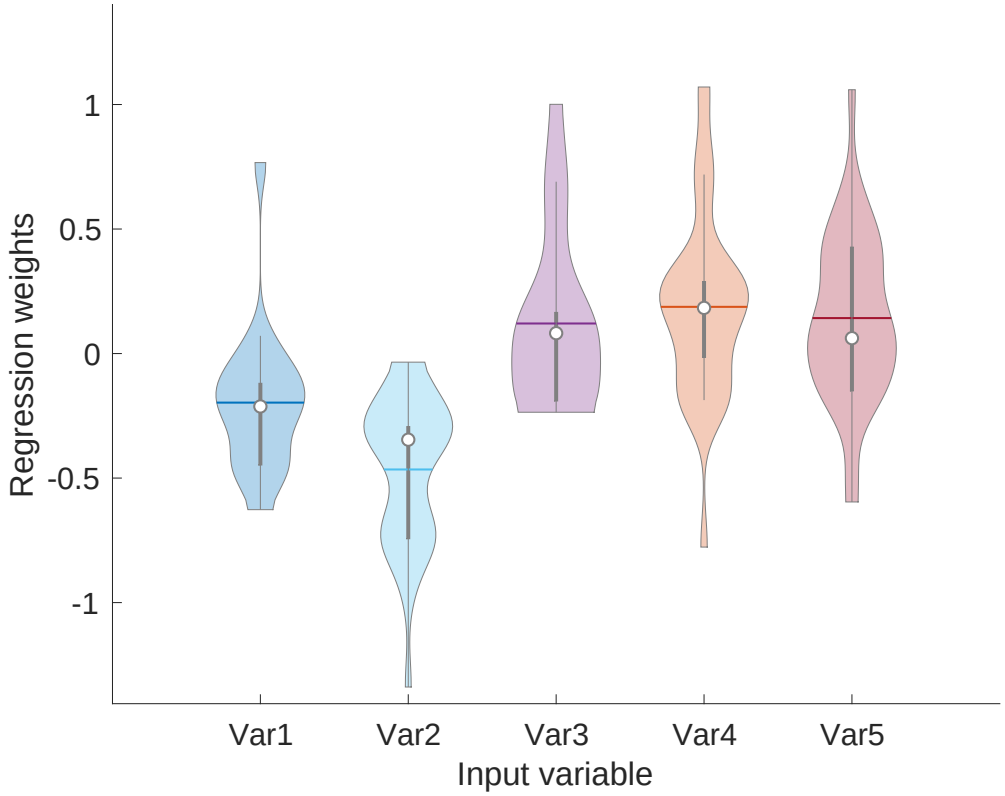

Figure 3.12: Violin plot CIR weights for the Real Estate data set.

Yacht data has just 308 observations—which is less than 6!=720 possible sort orders for a 6-dimensional data—the trained lattice did not cover all possible paths, and we could notice several nodes that are not connected to the rest of the lattice. In this scenario of limited training data, we could also notice that the ℓ_2 regularization had a visible impact on most of the larger nodes.

The heatmap of interaction indices and the violin plot of regression weights in Figures. 3.16 and 3.15, respectively, reveal that only the variables *Var1*, *Var3*, and *Var6* are the significant contributors to the regression outputs, while the rest are indifferent. The violin plots of the variables 2, 4, and 5 have a narrow vertical spread with a close-to-zero mean value.

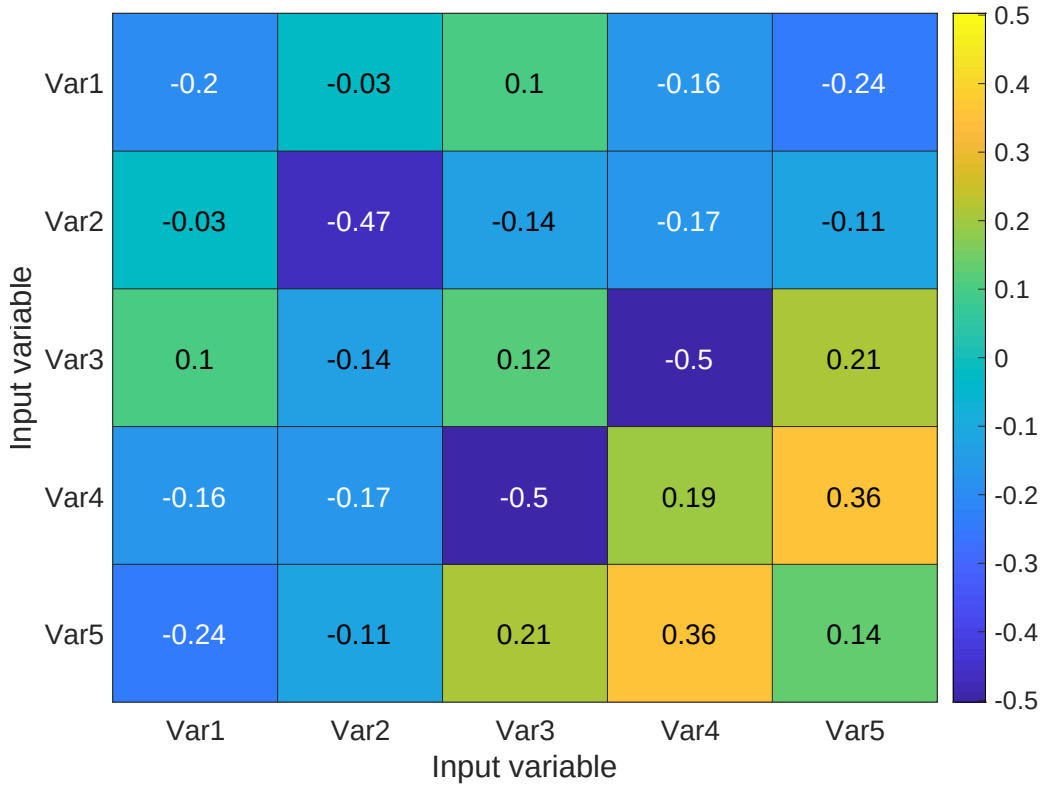

Figure 3.13: Heatmap of interaction indices for the Real Estate data set.

Thus, making them inconsequential in the regression solution. With these insights from the visualizations, for this data set, we could consider re-training the model using only the variables 1, 3, and 6. By removing the redundant variables, the new BC lattice will now have only 2^3=8 parameters—making it a more manageable learning problem with a limited training data.

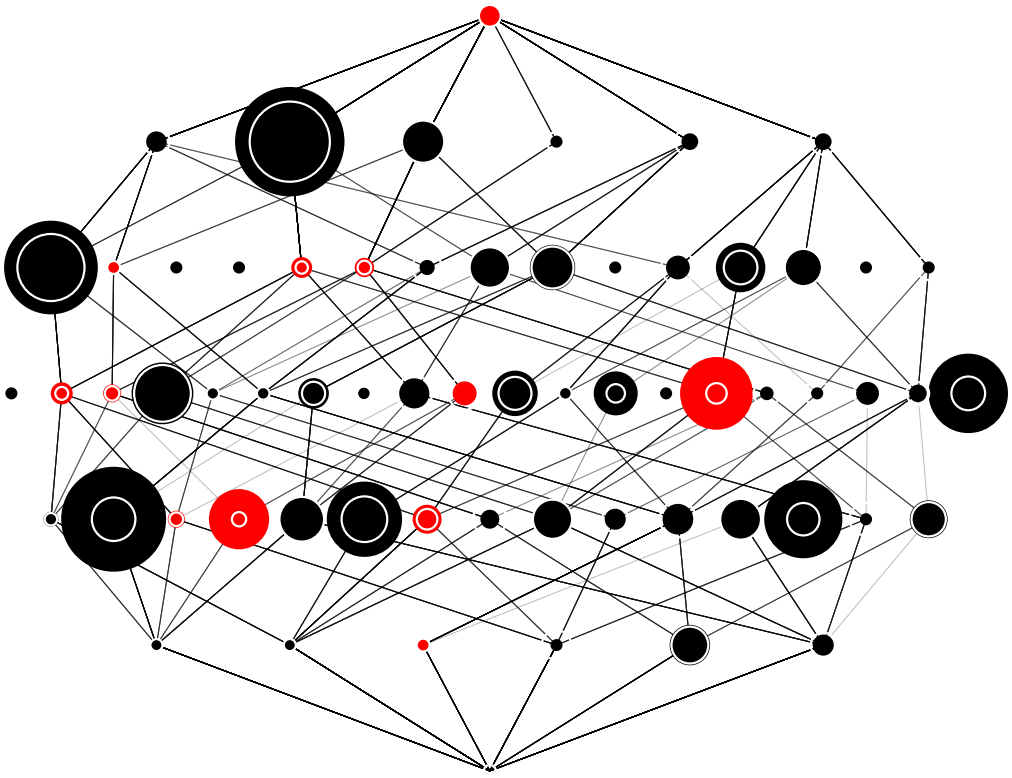

Figure 3.14: BC lattice of the CIR model trained on the Yacht data set—ℓ_2 regularization parameter value= 0.0001.

3.3.6 Conclusions

We reviewed our previously developed method of learning regression models using the Choquet fuzzy integral [59]. We then presented the evaluation indices that process the learned BC to quantitatively summarize the significance of individual variables, and assess the interactions between them. We presented visualization strategies to study the learned BC lattice, to tease apart the interactions among the input variables, and to holistically

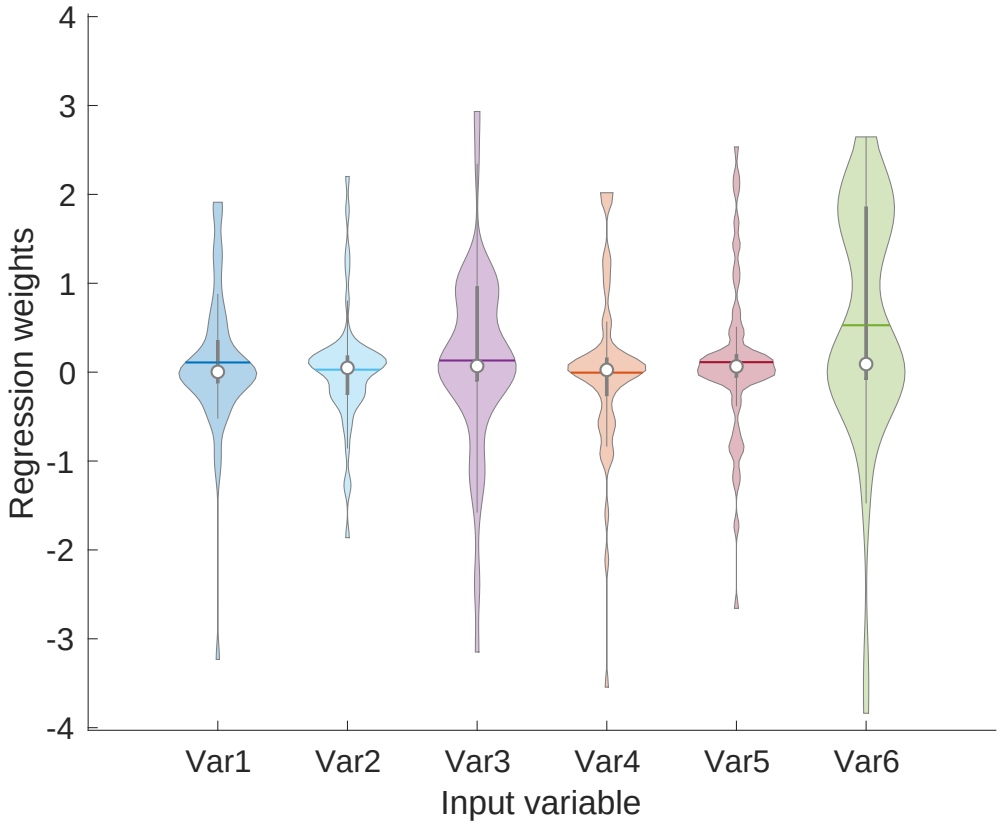

Figure 3.15: Violin plot CIR weights for the Yacht data set.

examine the distribution of regression weights. We applied these strategies on four benchmark regression data sets, and demonstrated how they can be leveraged in explaining the CIR model, and to make possible improvements to the models.

Figure 3.16: Heatmap of interaction indices for the Yacht data set.

Chapter 4

Online learning of the Fuzzy Choquet Integral for Feature-level fusion

4.1 Introduction

Chapter 3 presented a QP-based method (i.e., batch method) for learning the CIR. For higher dimensional data sets, i.e., $d > 6$, where d is the number of input features, the space complexity is so high that it becomes impractical to learn the CIR using the batch approach. This has limited the practical applicability of our ChI-based regression models to data sets with fewer dimensions. Alternatively, we applied dimensionality reduction methods like PCA to reduce the data to a manageable number of dimensions before applying the batch

approach, which resulted in loss of information and possibly sub-optimal models. In this chapter, we introduce an online learning algorithm to learn CIR models. This is a gradient descent approach in which we sequentially process each observation in the training data and adjust the parameters of the FM based on the gradient at that point. With this approach, we were able to extend the application of CIR to data sets of much larger dimensionality. We have also evaluated the application of ℓ_1 and ℓ_2 [69] regularization on our online learning method.

4.2 Online Learning of the CIR

Consider the minimization of the SSE presented at (3.12). This equation is the summation of the squared error calculated over all the observations in the training data. The corresponding squared error for a single data point (D_i, y_i) is

$$E_{Di}^2 = (D_i\mathbf{u} - y_i)^2 . \tag{4.1}$$

Differentiating (4.1) with respect to $\mathbf{u}$ gives

$$\nabla_{\mathbf{u}}(E_{Di}^2) = 2\,(D_i\mathbf{u} - y_i)\,D_i^T . \tag{4.2}$$

Now we update the BC vector $\mathbf{u}$ as

$$\mathbf{u} = \mathbf{u} - \gamma \left(D_i \mathbf{u} - y_i \right) D_i^T, \tag{4.3}$$

where γ is the learning rate of the gradient descent.[1] The iterative algorithm for training the BC vector $\mathbf{u}$ using the gradient descent approach is described in Algorithm 1.

Algorithm 1: Online Learning of CIR

Data: (D, y, γ, n) - Training data, output vector, learning rate, and number of training epochs
Result: $\mathbf{u}$ - Lexicographically ordered BC vector
Initiate the BC vector $\mathbf{u}$ with zeros.
for *epochs 1 to n* **do**
    for *each observation* $D_i \in D$ **do**
        Update the BC vector $\mathbf{u}$ as
        $\mathbf{u} = \mathbf{u} - \gamma \left(D_i \mathbf{u} - y_i \right) D_i^T$.
Output the learned BC vector $\mathbf{u}$.

4.2.1 ℓ_2 regularized online-CIR

We modify the SSE cost function at (3.12) to include the ℓ_2 penalty on the magnitude of the BC vector, yielding

$$E_{\ell_2}^2 = \sum_{i=1}^{n} (D_i \mathbf{u} - y_i)^2 + \lambda \|\mathbf{u}\|_2^2, \quad \lambda \geq 0, \tag{4.4}$$

[1] Note that constant factor of 2 present in Equation (4.2) was dropped in the update equation at (4.3).

where λ is the regularization parameter defining the weight of the regularizer: a user-tuned quantity. The corresponding squared error for a single data point (D_i, y_i) is

$$E^2_{\ell_2 D_i} = (D_i\mathbf{u} - y_i)^2 + \lambda\|\mathbf{u}\|_2^2, \tag{4.5}$$

The gradient of $E_{\ell_2 D_i}$ is calculated by differentiating (4.5) with respect to $\mathbf{u}$,

$$\nabla_{\mathbf{u}}(E^2_{\ell_2 D_i}) = 2\left(D_i\mathbf{u} - y_i\right) D_i^T + 2\lambda\mathbf{u}. \tag{4.6}$$

Thus the BC vector update equation for the ℓ_2 regularized CIR is

$$\mathbf{u} = \mathbf{u} - \gamma((D_i\mathbf{u} - y_i)\, D_i^T + \lambda\mathbf{u}). \tag{4.7}$$

Because $\lambda > 0$, at each iteration of the update equation at (4.7), in addition to adjusting the BC vector $\mathbf{u}$ to minimize the error, it also decreases the magnitude of individual parameters of the vector $\mathbf{u}$ by a small proportion. The iterative algorithm for training the ℓ_2 regularized BC vector $\mathbf{u}$ is presented in Algorithm 2.

Algorithm 2: Online Learning of the ℓ_2 regularized CIR

Data: $(D, y, \gamma, n, \lambda)$ - Training data, output vector, learning rate, number of training epochs, and regularization parameter λ

Result: $\mathbf{u}$ - Lexicographically ordered BC vector

Initiate the FM vector $\mathbf{u}$ with zeros.

for *epochs 1 to n* **do**

  for *each observation* $D_i \in D$ **do**

    Update the BC vector $\mathbf{u}$ as

    $\mathbf{u} = \mathbf{u} - \gamma \left((D_i\mathbf{u} - y_i)D_i^T + \lambda\mathbf{u} \right)$.

Output the learned BC vector $\mathbf{u}$.

4.2.2 ℓ_1 regularized Online Learning of the CIR

Applying the ℓ_1 penalty on the SSE cost function at (3.12) yields

$$E_{\ell_1}^2 = \sum_{i=1}^{n} (D_i\mathbf{u} - y_i)^2 + \lambda\|\mathbf{u}\|_1, \quad \lambda \geq 0, \tag{4.8}$$

where λ is the ℓ_1 regularization parameter. For a single data point (D_i, y_i), the corresponding ℓ_1 regularized squared error is

$$E_{\ell_1 D_i}^2 = (D_i\mathbf{u} - y_i)^2 + \lambda\|\mathbf{u}\|_1, \tag{4.9}$$

The gradient of the ℓ_1 regularized squared error with respect to $\mathbf{u}$ is

$$\nabla_{\mathbf{u}}(E_{\ell_1 D_i}^2) = 2\left(D_i\mathbf{u} - y_i\right) D_i^T + \lambda \text{sgn}(\mathbf{u}), \tag{4.10}$$

where the function sgn($\mathbf{u}$) returns the sign of the individual components of the vector $\mathbf{u}$. Using this gradient, the corresponding update equation for the ℓ_1 regularized CIR is

$$\mathbf{u} = \mathbf{u} - \gamma((D_i\mathbf{u} - y_i)\, D_i^T + \lambda \text{sgn}(\mathbf{u})). \tag{4.11}$$

Thus, with each iteration, in addition to adjusting the BC vector in favor of a lower SSE, Equation (4.11) also reduces the magnitude of the individual components of the BC vector $\mathbf{u}$ by the static value λ. The complete iterative algorithm for learning the ℓ_1 regularized CIR is presented in Algorithm 3.

Algorithm 3: Online Learning of the ℓ_1 regularized CIR

Data: $(D, y, \gamma, n, \lambda)$ - Training data, output vector, learning rate, number of training epochs, and regularization parameter λ
Result: $\mathbf{u}$ - Lexicographically ordered BC vector
Initiate the BC vector $\mathbf{u}$ with zeros.
for *epochs 1 to n* **do**
  for *each observation* $D_i \in D$ **do**
    Update the BC vector $\mathbf{u}$ as
    $\mathbf{u} = \mathbf{u} - \gamma\left((D_i\mathbf{u} - y_i)D_i^T + \lambda \text{sgn}(\mathbf{u})\right)$.
Output the learned BC vector $\mathbf{u}$.

Table 4.1
Three Methods for Building Bias Vector β [59].

Name	Description	B_i
1-bias	One bias term	$B_i = 1, \forall i$
d-bias	One term for each max-value in sort	$[B_i]_{\pi(1)} = 1$ else $[B_i] = 0$
2^d-bias	Computed bias for each possible sort	$[B_i]_j = \begin{cases} 1 & [H_i]_j > 0, \\ 0 & \text{else.} \end{cases}$

4.3 Experiments

We evaluated our online method for learning the CIR using real world data sets from the UCI Machine Learning repository [33] and compared the performance with our previously proposed QP-based CIR learning method (which we call batch-CIR). We performed 100 experimental trials on each method-data set pair—in each trial, a random permutation of the data set is split into 75%/25% for training and testing, respectively. For the Online-CIR method, the iterative training involved 100 epochs[2]. We chose 100 since no significant improvement in model performance was observed beyond 50 training epochs on most data sets. We z-normalized all the features as well the target variable to have a zero mean and a unit standard deviation prior to training the algorithms. We also tested the ℓ_1 and ℓ_2 regularized online learning methods of CIR described in Algorithms 2 and 3. Table 4.2 shows the performance comparison. The CIR β models are indicated by CIR(b), and the CIR-online models are indicated by CIR(b)-online, where b indicates the number of bias terms $\{1, d, 2^d\}$—see Table 4.1.

[2]An epoch is a single cycle of training through all the observations in the training data.

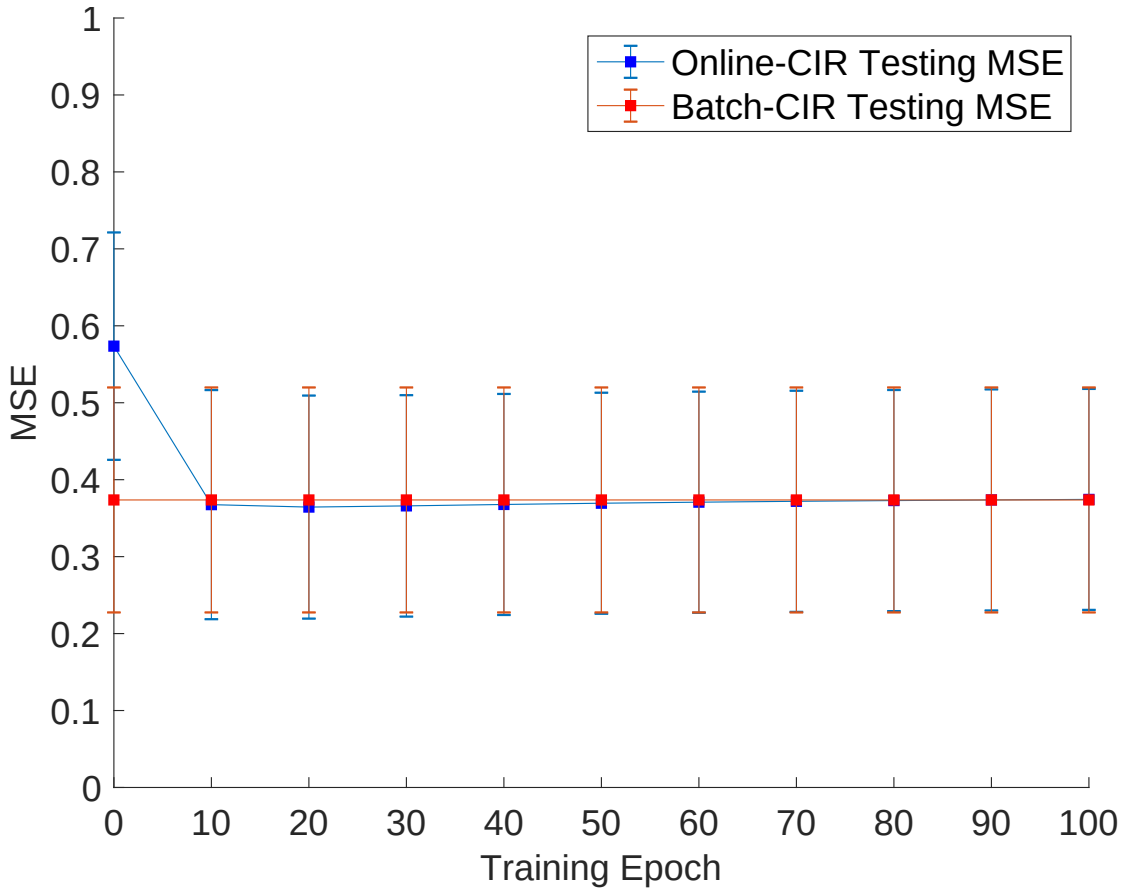

Figure 4.1: CIR(1)-online vs. CIR(1) batch learning—Performance on RealEstate data set ($d = 5$). The mean MSE observed over 100 experimental trials—online method is trained for 100 epochs in each trial. The error bars indicate the width of one standard deviation on both sides of the mean MSE.

4.3.1 Performance on low-dimensional data sets

Figure 4.1 shows the typical learning trend we observed on all the low-dimensional data sets ($d \leq 6$). The MSE of online-CIR has matched the performance of the batch-CIR within 40-50 epochs. In some cases, like the one shown in Figure 4.1, we noticed that the error rate of online-CIR dipped slightly below that of batch-CIR initially, but converged to

the batch-CIR MSE after several more epochs of training.

4.3.2 Performance on high-dimensional data sets

The online-CIR method consistently outperformed batch-CIR on data sets with seven or more dimensions. Among the 10 real-world data sets we used for testing, six had seven or more dimensions. For these high-dimensional data sets, since it was computationally impractical to directly apply our batch-CIR approach, we applied *principal component analysis* (PCA) to reduce the dimensionality to 6-dimensions prior to training. For the online-CIR method, we were able to use all the data sets without any dimensionality reduction. The online-CIR method achieved a significantly lower MSE on 5 of these 6 data sets. Table 4.2 shows the performance comparison. We performed a two-tailed t-test at a 5% statistical significance to identify the statistically best results on each data set. Except for the Red wine data set, the online-CIR method showed superior performance on all the high-dimensional data sets ($d > 6$).

Figure 4.2 compares the performance of our online-CIR method with the batch-CIR method on the Concrete data set. The online-CIR method achieved a lower MSE than the batch-method in just 20 epochs through the data. After 100 epochs, the MSE of the CIR(1)-online was 0.199, significantly lower than the best MSE achieved from the batch method, CIR(d), at 0.322. This trend was consistently observed on 5 out of the 6 high-dimensional data sets,

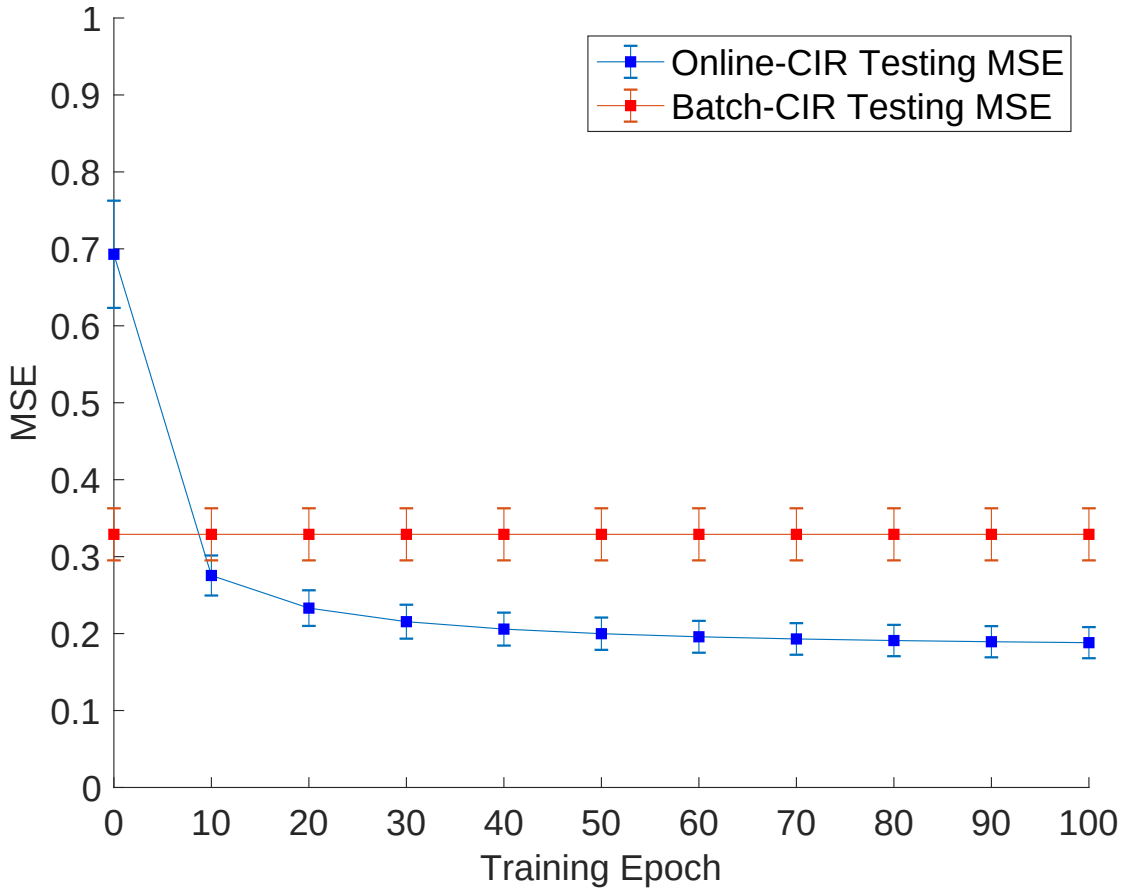

Figure 4.2: CIR(1)-online vs. CIR(1) batch learning—Performance on Concrete data set. The online method was trained using the original data set ($d = 8$), and the batch method was trained using the PCA-reduced 6D-data set.

where the online-methods outperformed the batch-methods.

Because we applied PCA to reduce the training data of batch-CIR methods for data sets with 7 or more dimensions to 6D, we conjecture that the poorer performance of the batch-CIR methods was likely due to the loss of information from the dimensionality reduction. To test this, we ran the online-CIR method on the PCA-reduced data sets and compared the results with the batch-CIR method. Figure 4.3 shows the performance comparison of the online and the batch methods on the PCA-reduced 6D-Concrete data set. The MSE of the

online-CIR converged to equivalent performance as the batch-method after several training epochs. This trend was consistently observed on all the data sets with 7 or more dimensions on which the PCA-reduction was applied. While this indicates that the online-CIR method does not provide any performance boost relative to the batch-CIR methods, this is still a positive result, since the results of batch- and the online-CIR methods showed equivalent performance when the same training data were used. However, the online-CIR method can be used to extend the application of the CIR methods to higher-dimensional data sets with a performance matching the hypothetical performance that batch-CIR method could achieve (absent computational issues with high-dimensional data).

4.3.3 Convergence time

Our experimental results have demonstrated that both the online- and batch-CIR methods produce identical performance. However, since the training process of the online-CIR method is iterative, we evaluated the time taken by the online-methods to converge with the performance of the batch-methods. For a training data set, the online-CIR method is considered to have converged with the batch-method when the test-MSE of at least one of the online-CIR(b) methods comes within 5% of the best MSE observed with the batch-CIR(b) methods. Figure 4.4 shows the number of epochs taken by the online-CIR(b) methods to converge to the batch-CIR(b) methods on each data set. While one would expect the high-dimensional data sets in general to take more epochs to converge, we have not noticed

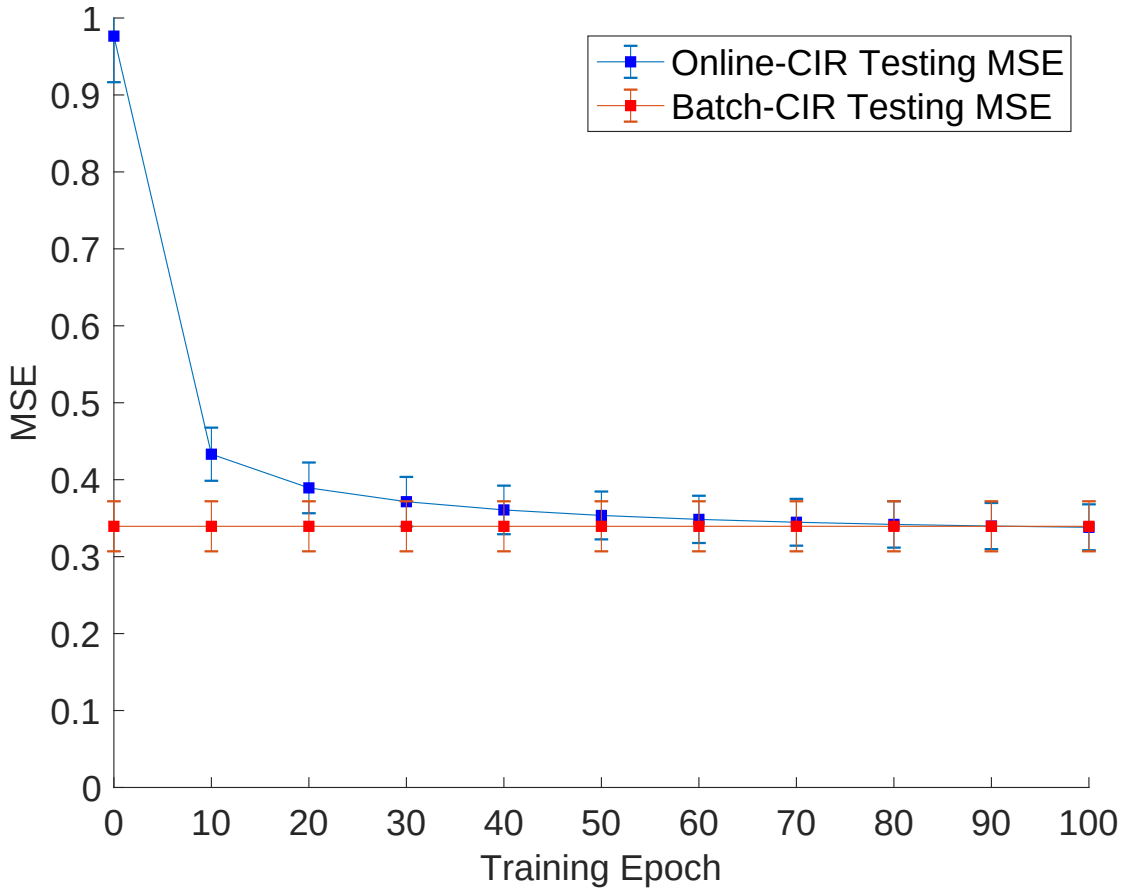

Figure 4.3: CIR(1)-online vs. CIR(1) batch learning—Performance on 6D-Concrete data set. Both the online and the batch methods converged to the same error rate when using the same PCA-reduced 6D-training data.

any such strong correlation. Seven of the 10 data sets showed convergence in less than 15 epochs, and the 3 data sets—ENB2 ($d = 8$), Yacht ($d = 5$) and Airfoil ($d = 5$)—that took more than 40 epochs to converge are not particularly high-dimensional relative to the rest of the data sets.

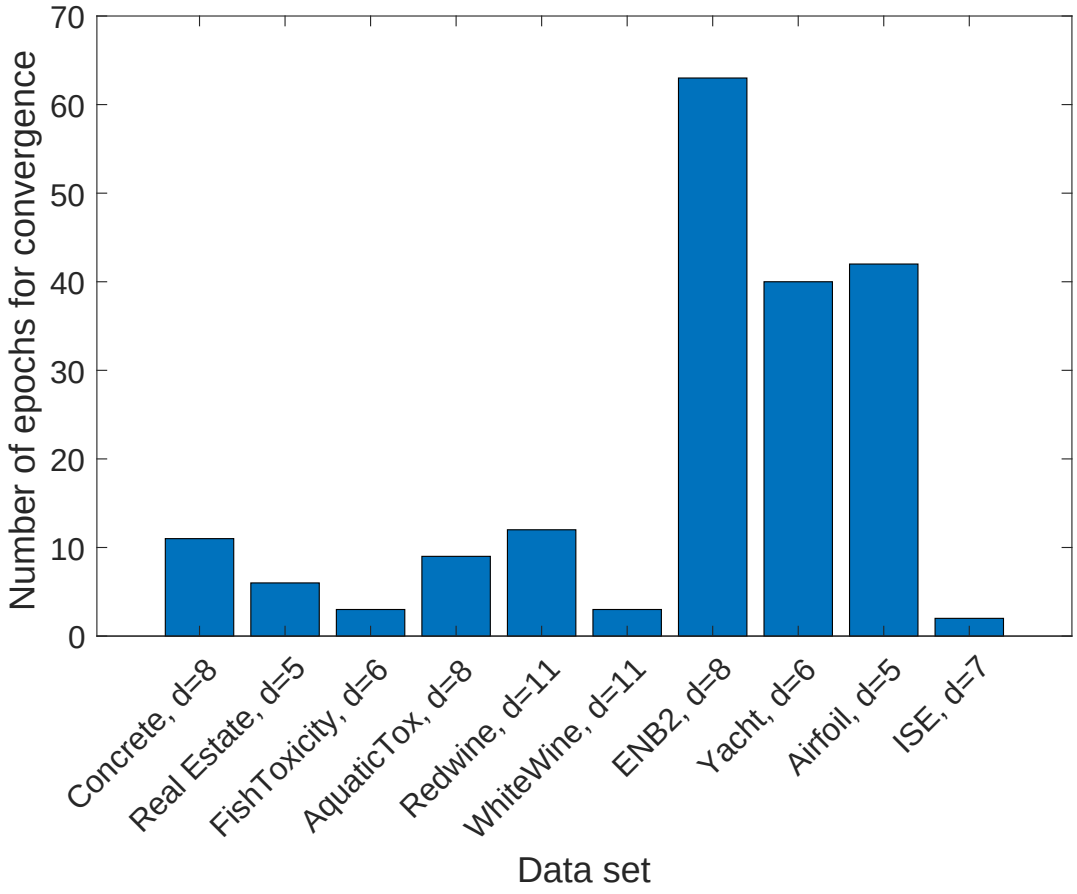

Figure 4.4: Number of epochs for convergence of the MSE of at least one of the online-CIR(b) methods with Batch-CIR.

4.3.4 Impact of ℓ_1 and ℓ_2 regularization on online-CIR

We trained the ℓ_2- and ℓ_1-regularized online-CIR(b) methods as per the Algorithms 2 and 3, respectively. Though we observed some instances where a regularized online-CIR method produced a slightly lower MSE compared to the base online-CIR methods, the improvement was not statistically significant. On all the data sets with a regularized online-CIR algorithm among statistically-best performing methods, we also have a base online-CIR

algorithm (with no regularization) with a statistically similar performance as identified by the two-tailed t-test at a 5% significance level. In other words, regularization did not show statisitcally significant performance enhancement in these experiments.

4.3.5 Comparison with deep neural networks

To compare the performance of CIR methods with *deep neural networks* (DNNs), we built 3-hidden layer DNNs on the UCI regression data sets. Each hidden layer comprised 256 nodes. We performed 10 experimental trials—in each trial, a random permutation of the data set is split into 75%/25% for training and testing, respectively. The DNNs were trained for 40 epochs in each experiment. The performance of DNNs is presented in Table 4.2. On 7 out of 10 data sets, the DNN model has either matched or outperformed the CIR methods. While this indicates the superior performance of DNNs, a DNN model for a 6-dimensional data set comprises 132,864 learned weights, making it impractical to explain what was learned by the model or the outputs it produces. In contrast, the CIR model for a 6-dimensional data set comprises just 64 trainable parameters. In addition, the visualization methods and evaluation metrics introduced in this chapter enable the explainability of the model and the results produced. So the choice of using DNNs vs. CIR methods depend on requirements of the user. For situations where explainability of the solutions is important, the CIR models provide it with a relatively minimal loss in performance compared to the DNNs.

Table 4.2
MSE on Benchmark Data Sets*

Method	Concrete	Real Estate	Fish Toxicity	Aquatic Toxicity	Red Wine	White Wine	ENB-2	Yacht	Airfoil	ISE	# of best instances
n	1,030	414	908	546	1599	4898	768	308	1503	536	-
d	8	5	6	8	11	11	8	6	5	7	-
CIR(1)	0.322 (0.034)	**0.365 (0.144)**	**0.426 (0.061)**	0.636 (0.115)	**0.665 (0.060)**	0.792 (0.067)	0.055 (0.007)	**0.155 (0.034)**	0.357 (0.037)	0.546 (0.081)	4
CIR(d)	0.312 (0.032)	**0.367 (0.135)**	**0.429 (0.056)**	0.625 (0.109)	**0.7 (0.070)**	0.801 (0.089)	0.052 (0.005)	**0.157 (0.038)**	**0.342 (0.037)**	0.560 (0.071)	5
CIR(2^d)	0.324 (0.033)	**0.349 (0.123)**	0.493 (0.086)	0.906 (0.383)	0.737 (0.066)	0.809 (0.072)	0.095 (0.422)	**0.224 (0.105)**	**0.315 (0.031)**	0.680 (0.094)	3
CIR(1)-online	**0.199 (0.021)**	**0.368 (0.149)**	**0.436 (0.056)**	**0.521 (0.072)**	**0.679 (0.061)**	**0.709 (0.052)**	0.047 (0.007)	**0.170 (0.040)**	0.360 (0.030)	**0.479 (0.062)**	8
CIR(d)-online	**0.199 (0.021)**	**0.379 (0.144)**	**0.451 (0.065)**	**0.537 (0.082)**	**0.703 (0.075)**	0.797 (0.075)	0.045 (0.007)	**0.185 (0.056)**	0.357 (0.034)	**0.481 (0.062)**	7
CIR(2^d)-online	0.217 (0.030)	**0.363 (0.145)**	**0.474 (0.061)**	0.809 (0.152)	1.376 (0.159)	1.487 (0.144)	**0.019 (0.004)**	**0.167 (0.045)**	**0.345 (0.036)**	0.584 (0.075)	5
CIR(1)-online-ℓ_1	**0.201 (0.021)** λ =0.00001	**0.345 (0.138)** λ =0.01	**0.424 (0.061)** λ =0.0001	**0.518 (0.085)** λ =1	**0.671 (0.063)** λ =0.0001	**0.740 (0.069)** λ =1	0.047 (0.008) λ =0.01	**0.157 (0.037)** λ =0.001	0.366 (0.037) λ =0.001	**0.459 (0.064)** λ =0.1	8
CIR(d)-online-ℓ_1	**0.191 (0.023)** λ =0.0001	**0.357 (0.143)** λ =1	**0.431 (0.058)** λ =0.001	**0.513 (0.078)** λ =1	**0.707 (0.069)** λ =0.1	**0.737 (0.074)** λ =0.0001	0.045 (0.007) λ =0.001	**0.174 (0.044)** λ =0.0001	0.357 (0.035) λ =0.00001	**0.476 (0.065)** λ =0.001	8
CIR(2^d)-online-ℓ_1	**0.206 (0.021)** λ =1	**0.332 (0.134)** λ =1	**0.462 (0.060)** λ =1	**0.563 (0.081)** λ =1	0.846 (0.097) λ =1	0.777 (0.072) λ =1	**0.018 (0.004)** λ =0.1	**0.163 (0.046)** λ =0.00001	**0.331 (0.038)** λ =0.001	0.525 (0.068) λ =1	7
CIR(1)-online-ℓ_2	**0.195 (0.023)** λ =0.0001	**0.345 (0.144)** λ =1	**0.432 (0.059)** λ =0.001	**0.515 (0.072)** λ =0.1	**0.677 (0.058)** λ =0.1	**0.695 (0.052)** λ =0.1	0.047 (0.006) λ =0.00001	**0.162 (0.039)** λ =0.01	0.349 (0.030) λ =1	**0.467 (0.061)** λ =1	8
CIR(d)-online-ℓ_2	**0.195 (0.019)** λ =0.01	**0.350 (0.129)** λ =1	**0.440 (0.060)** λ =0.001	**0.521 (0.082)** λ =0.0001	**0.667 (0.061)** λ =0.0001	0.773 (0.074) λ =0.001	0.046 (0.007) λ =0.0001	**0.177 (0.043)** λ =0.001	0.351 (0.034) λ =0.001	**0.473 (0.063)** λ =0.1	7
CIR(2^d)-online-ℓ_2	**0.201 (0.021)** λ =1	**0.345 (0.138)** λ =1	**0.468 (0.060)** λ =1	0.640 (0.094) λ =1	0.902 (0.080) λ =1	0.944 (0.082) λ =1	**0.019 (0.003)** λ =1	**0.168 (0.049)** λ =0.01	**0.338 (0.038)** λ =0.00001	0.545 (0.070) λ =1	6
Deep_learning** (132,864 parameters)	***0.139 (0.018)***	0.395 (0.145)	***0.416 (0.041)***	***0.488 (0.059)***	***0.689 (0.089)***	0.772 (0.084)	0.043 (0.012)	***0.031 (0.013)***	***0.17 (0.027)***	***0.491 (0.049)***	7

*MSE values in the table are the average testing MSE of 100 experimental trails. The corresponding standard deviation is enclosed in parentheses. **Deep learning model comprised of 3 hidden layers with 256 nodes each. Cells marked in bold-italic indicate data instances for which the deep learning model has either matched or outperformed the ChI-based models.
Bold (non-italicized) indicates lowest MSE at a 5% statistical significance.

4.4 Conclusion

This chapter introduced an online method for learning the CIR. The online method has relaxed the space complexity problem of our previously proposed QP-based batch learning method. The experimental evaluation on real-world data sets demonstrated that our online-CIR method outperformed the batch-method on high-dimensional data sets (since it does not require dimensionality reduction), and matches the performance on the low-dimensional data sets.

Chapter 5

Novel Regularization for Learning the Fuzzy Choquet Integral with Limited Training Data

5.1 Introduction

Classification problems generally seek a function, $f(\cdot)$, that can transform or map an observation, $\mathbf{x}$, to a prediction or decision, $y = f(\mathbf{x})$. Machine learning is often utilized to determine a suitable function $f(\cdot)$ from a set of training data $(\mathbf{y}, X)$, where $\mathbf{y} = (y_1, y_2, ..., y_n)^T$ (a vector of labels) and $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ (a set of feature vectors). Learning the

prediction function f typically involves solving an optimization problem that minimizes the error between the true labels of the training data and the labels predicted by the function f.

Linear classifiers are a commonly used classification approach where a classification decision is made based on a linear combination of the input feature-vector data. For binary classification, this is equivalent to finding a hyperplane in the feature-space that separates the two classes of training data with minimal error; this hyperplane is also known as the decision boundary. There are many linear classifiers that all differ in the way they identify the decision boundary, however, work presented in this chapter utilizes *support vector machines* (SVMs) [29].

SVMs are popular hyperplane classifiers that are easy to train and computationally efficient. However, SVMs (and other linear classifiers) require the data to be linearly separable: a property that is not frequently encountered with real data. The kernel-based variants of SVMs work around this problem by projecting the data into a higher-dimensional space where the data are linearly separable. While this seems to solve the problem of non-separable data, we are still faced with the problem of identifying the appropriate kernel function that yields the desired projection, which is not trivial.

Multiple kernel learning (MKL) solves this problem by learning a new kernel as a combination of predetermined base kernels while maintaining the necessary properties of a kernel (positive-semidefiniteness). This approach is discussed in many works

[28, 64, 74, 97, 101, 126]. MKL combines different "perpectives" of the data (each represented by a separate kernel) into a single classifier to determine the predicted class label. A different MKL method uses multiple kernel-based classifiers wherein each uses a different kernel, and an aggregation function combines the outputs of the individual classifiers. In the DeFIMKL approach introduced in [97], the aggregation is performed using the ChI with respect to an FM. This method, however, still presents a challenge: how do we specify the FM?

This chapter addresses the open issue of specifying the FM. In particular, we focus on learning parts of the FM that are not learned from training data. As we will illuminate later, learning the entire FM in a data-supported manner is typically impossible with real data sets; only a subset of the FM is accurately learned from the training data. Therefore when given a new instance to classify (a testing datum) the model runs the risk of using parts of the FM that are not data-supported, possibly leading to an erroneous prediction.

In previous work [99] we proposed a regularization-based method that allows the user to specify a goal for the learned FM such that parts of the learned FM that are not data-supported are assigned reasonable values. The goals we previously explored were minimum, maximum, and mean, that is, using a regularization function we pushed the learned FM towards a min-like FM, max-like FM, or mean-like FM. These regularizations were implemented using ℓ_2-norms. The work presented in this chapter builds upon this prior work by extending these regularizers to the ℓ_1-norm case and present algorithms for their

solutions. Furthermore, this chapter extends the concept of pushing the learned FM to a goal, where the goal is *not* known *a priori*. This extension can also be interpreted as constraining the learned FM to achieve a particular structure; here, we constrain the learned FM to resemble an *ordered weighted average* (OWA).

Finally, an additional contribution of this work is the exploration of the algorithms' utility across a more comprehensive set of experiments using both synthetic and real-world data sets. The novel algorithms' performance is compared against each other as well as against other state-of-the-art classification algorithms.

5.2 Goal-based regularization strategies for learning the ChI

5.2.1 Common aggregations via the Choquet FI

The ChI is capable of representing many aggregation functions [113]. For example, the Choquet integral acts as a minimum operator when the FM g is all 0s (except $g(X) = 1$, due to boundary constraints), the mean operator when $g(A_i) = |A_i|/m, \forall A_i \subset X$, and the maximum operator when the FM is all 1s (except $g(\emptyset) = 0$, due to boundary constraints).

5.2.2 Training The DeFIMKL Algorithm

The DeFIMKL algorithm is a method of decision level fusion introduced in [97]. This algorithm uses the Choquet FI to non-linearly fuse the decisions from an ensemble of classifiers. The mathematical description of this algorithm was presented in Section 1.3 of Chapter I.

The QPs at (1.9) and (1.10) provide a method to learn the FM $\mathbf{u}$ (i.e., g) from training data. 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 k^{th} SVM classifier's decision value is

$$\eta_k(\mathbf{x}) = \sum_{i=1}^{n} \alpha_{ik} y_i \kappa_k(\mathbf{x}_i, \mathbf{x}) - b_k, \tag{5.1}$$

which is interpreted as the signed distance of $\mathbf{x}$ from the hyperplane defined by the learned SVM model parameters, α_{ik} and b_k [15, 30]. The class label is typically computed as $\text{sgn}\{\eta_k(\mathbf{x})\}$,[1] which could be used as the training input to the FM learning at (1.6); however, we remap $\eta_k(\mathbf{x})$ onto the interval $[-1, +1]$ via the sigmoid function to create inputs for

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

learning as

$$f_k(\mathbf{x}) = \frac{\eta_k(\mathbf{x})}{\sqrt{1+\eta_k^2(\mathbf{x})}}. \tag{5.2}$$

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 κ_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 summarizes the training process. After training, a new feature vector $\mathbf{x}$—from a test data set—can be classified via the procedure summarized in Algorithm 5.

Algorithm 4: DeFIMKL Classifier Training [99]

Data: $(\mathbf{x}_i, y_i)$ - feature vector and label pairs; K_k - kernel matrices
Result: $\mathbf{u}$ - Lexicographically ordered FM vector
for *each kernel matrix* **do**
    Compute the kernel SVM classifier decision values, η_k, as in (5.1).
    Remap the decision values onto the interval $[-1, +1]$ as f_k using (5.2).
Solve the minimization problem in (3.7) for the FM $\mathbf{u}$.

Algorithm 5: DeFIMKL Classifier Prediction [99]

Data: $\mathbf{x}$ - feature vector; K_k - kernel matrices; $\mathbf{u}$ - learned fuzzy measure vector
Result: y - Predicted class label
Compute the SVM decision values $f_k(\mathbf{x})$ by using (5.1) and (5.2).
Apply the Choquet integral at (1.2) with respect to the learned FM $\mathbf{u}$.
Compute the class label as $y = \text{sgn}\{C_g(\mathbf{x})\}$.

5.2.3 Learning the FM with insufficient training data

The DeFIMKL classifier for m classifiers would require an FM with 2^m (or $2^m - 2$, observing the boundary conditions in property P4) variables. Therefore, we at least need as many observations to train such an FM. Since the real-world data sets are not very likely to be comprised of so many unique sort orders, there will likely be values of the FM that are not data-supported. The values for such nodes are set based on the monotonicity constraints in the QP, the type of initialization used in the QP solver, and the choice of the regularization used. It is therefore highly likely that the learned values do not accurately represent the underlying FM, and when a new data point that uses one or more of such untouched nodes is fused using Algorithm 5, the prediction accuracy will suffer. The behavior of the ℓ_2-regularized DeFIMKL algorithm when presented with insufficient training data is demonstrated in the following contrived example.

Example 4. *Learning an Underdetermined FM via ℓ_2-regularized DeFIMKL [99].* We trained an ℓ_2-regularized DeFIMKL algorithm with three-SVMs (i.e., $m = 3$, however, without loss of generality these results are also indicative of the behavior when $m > 3$). Using the underlying FM (FM was arbitrarily assigned) shown in Table 5.1, we generated a synthetic training data set that purposefully avoids two nodes in the fuzzy lattice. The DeFIMKL algorithm learned the FM using this synthetic data; Table 5.1 shows the FM learned for different regularization choices with $\lambda = 1$. Note that two nodes in the lattice,

$g(\{x_2\})$ and $g(\{x_1, x_2\})$, are not data-supported and thus are driven by the monotonicity constraints and the regularization choice.

In Table 5.1, the third column corresponds to ℓ_2-min regularization—here, we see that the nodes that are touched by the training data (i.e., nodes traversed by the Choquet integral) are learned successfully with minimal error (less than 5%). However for the two nodes that are untouched by the training data, since the values are driven by the monotonicity constraints, the node $g(\{x_2\})$ 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 $g(\{x_1, x_2\})$ gets a value of 0.14 to satisfy the constraint $\max\{g(\{x_1\}), g(\{x_2\})\} \leq g(\{x_1, x_2\}) \leq g(\{X\})$. In both of these cases, the learned FM value is essentially the minimum value permitted by the monotonicity constraints. This is due to the ℓ_2-regularization of the DeFIMKL algorithm.

As highlighted by Table 5.1, the nodes that are frequently visited by the training data obtain values that are relatively stable, independent of the type of regularization. More specifically, these nodes directly contribute towards the squared error calculated as per (1.5a). Thus, the error minimization method fits the values of these nodes more in alignment with the training data. On the other hand, the nodes that are not supported by the training data do not significantly contribute towards the squared error and thus have the flexibility to vary, as long as they obey the monotonicity constraints of the FM. This phenomenon is discussed in more detail in Sections 5.2.4 and 5.2.5.

Table 5.1

Underlying and learned FMs. The learned FM terms marked with asterisks are not supported by the training data. Regularization labels indicate the type of norm employed and the aggregation goal.

		Goal Regularization							
FM Term	**Underlying**	ℓ_2**-min**†	ℓ_2**-max**	ℓ_2**-mean**	ℓ_1**-min**‡	ℓ_1**-max**	ℓ_1**-mean**	ℓ_2**-LOS**	ℓ_1**-LOS**
$g(\{x_1\})$	0.14	0.14	0.19	0.15	0.12	0.14	0.15	0.15	0.14
$g(\{x_2\})$*	0.29	0.00017	0.93	0.33	8.7e-9	0.86	0.33	0.29	0.22
$g(\{x_3\})$	0.43	0.43	0.44	0.44	0.43	0.43	0.43	0.43	0.43
$g(\{x_1, x_2\})$*	0.57	0.14	1	0.67	0.12	1	0.67	0.78	0.69
$g(\{x_1, x_3\})$	0.71	0.69	0.71	0.71	0.71	0.71	0.7	0.71	0.71
$g(\{x_2, x_3\})$	0.86	0.83	0.93	0.87	0.85	0.86	0.85	0.86	0.86
See section:		5.2.4.1			5.2.4.2			5.2.5.2	5.2.5.3

† Equivalent to ℓ_2-regularization on the FM vector directly, i.e., $v_*(\mathbf{u}) = \lambda\|\mathbf{u}\|_2$.

‡ Equivalent to ℓ_1-regularization on the FM vector directly, i.e., $v_*(\mathbf{u}) = \lambda\|\mathbf{u}\|_1$.

5.2.4 FM Learning with a Specified Goal

The standard DeFIMKL algorithm discussed in the previous section assumes that the structure of the underlying FM is not known, and thus no information regarding the underlying FM is encoded in the QP. If, however, the FM is partially known, the QP at (1.10) should include that information. To this end, we propose the regularization function

$$v_*(\mathbf{u}) = \lambda \|\mathbf{u} - \mathbf{g}\|_p, \tag{5.3}$$

where $\mathbf{g}$ represents a goal of what we expect the underlying FM to look like and p defines the norm type. The following sections describe the solution to the regularized problem with ℓ_2- and ℓ_1-regularization.

5.2.4.1 ℓ_2-goal regularization

Including the regularization function from (5.3) in the QP with $p = 2$ gives[2]

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda \|\mathbf{u} - \mathbf{g}\|_2^2, \tag{5.4}$$

[2]Note that we square the regularization term in this case for mathematical convenience; the problem remains convex.

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 (5.4) leads to

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \left(\hat{D} + 2\lambda I\right) \mathbf{u} + (\mathbf{f} - 2\lambda \mathbf{g})^T \mathbf{u}, \tag{5.5}$$

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 (5.3) is quadratic in $\mathbf{u}$; the minimization problem at (5.5) can be solved by a QP solver.

5.2.4.2 ℓ_1-goal regularization

The regularization function at (5.3) with $p = 1$ forces $\mathbf{u}$ to lie close to the goal $\mathbf{g}$ in the ℓ_1-sense. Including this regularization function in the QP gives

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T \hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda\|\mathbf{u} - \mathbf{g}\|_1, \tag{5.6}$$

however, this formulation cannot simply be reduced or combined in the objective function any further as done in the previous case of ℓ_2-regularization; the difference within the norm will not, in general, be non-negative. To address this, we move the regularization term to the constraints through the use of Tibshirani's iterative lasso algorithm; see Appendix 5.5 for a brief description of the method. Algorithm 6 describes the process in terms of this problem.

This algorithm solves the unregularized problem while iteratively updating its constraints to enforce sparsity in the difference $(\mathbf{u} - \mathbf{g})$; although, the problem in (5.32) must be reformulated as follows. Let us first lump $\mathbf{u}$ and $\mathbf{g}$ into a single long vector $\mathbf{w}$ as $\mathbf{w} = [\mathbf{u}^T \quad \mathbf{g}^T]^T \in \mathcal{R}^{2^{m+1}-2}$. The unregularized QP in (3.7) can then be rewritten as

$$\min_{\mathbf{w}} 0.5\mathbf{w}^T \hat{D}_{\mathbf{w}} \mathbf{w} + \mathbf{f}_{\mathbf{w}}^T \mathbf{w}, \quad C_{\mathbf{w}} \mathbf{w} \leq \mathbf{0}, \quad \mathbf{b}_l \leq \mathbf{w} \leq \mathbf{b}_r, \tag{5.7}$$

where

$$\begin{gathered} \hat{D}_{\mathbf{w}} = \begin{bmatrix} \hat{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{f}_{\mathbf{w}} = [\mathbf{f}^T \ \mathbf{0}^T]^T, \quad C_{\mathbf{w}} = [C \ \mathbf{0}], \\ \mathbf{b}_l = [(\mathbf{0}, 1)^T \ \mathbf{g}^T]^T, \quad \mathbf{b}_r = [\mathbf{1} \ \mathbf{g}^T]^T. \end{gathered} \tag{5.8}$$

Denote the vector of the sign of the differences at the ith iteration as

$$\boldsymbol{\delta_i} = \text{sign}(\mathbf{u}_i - \mathbf{g}), \tag{5.9}$$

and the matrix including all $\boldsymbol{\delta}$s from iteration 0 as

$$G^i = \begin{bmatrix} \boldsymbol{\delta_0}^T & -\boldsymbol{\delta_0}^T \\ \boldsymbol{\delta_1}^T & -\boldsymbol{\delta_1}^T \\ \vdots & \vdots \\ \boldsymbol{\delta_i}^T & -\boldsymbol{\delta_i}^T \end{bmatrix}, \tag{5.10}$$

such that multiplication of G_i^i, the ith row of G^i, with the vector $\mathbf{w}$ represents an ℓ_1-summation, e.g.,

$$G_i^i \mathbf{w}_i = \|\mathbf{u}_i - \mathbf{g}\|_1, \tag{5.11}$$

where $\mathbf{u}_i$ is the learned FM from the ith iteration of the algorithm. Note the regularization parameter is $t \propto 1/\lambda$, and the vectorized version is denoted as $\mathbf{t} = t\mathbf{1}$. The iterative lasso algorithm for this problem is given in Algorithm 6, which follows the notation presented in (5.35)–(5.37).

Algorithm 6: ℓ_1-Goal Regularization via Tibshirani's Lasso Algorithm

Data: The QP at (5.33) and regularization parameter t.
Result: $\mathbf{u}$ - Lexicographically ordered fuzzy measure vector (recovered from $\mathbf{w}$).
$i = 0$;
$\mathbf{w}_0 \leftarrow$ solve the unregularized QP in (5.33);
$\boldsymbol{\delta_0} \leftarrow$ find the sign vector of (5.35);
$G^0 \leftarrow$ add $\boldsymbol{\delta_0}$ to G as shown in (5.36);
while $G_i^i \mathbf{w}_i > \mathbf{t}$ **do**
    $i \leftarrow i + 1$
    $\mathbf{w}_i \leftarrow$ solve the QP in (5.33) with the additional constraint $G^{i-1}\mathbf{w}_{i-1} \leq \mathbf{t}$.
    $\boldsymbol{\delta_i} \leftarrow$ find the sign vector of (5.35).
    $G^i \leftarrow$ add $\boldsymbol{\delta_i}$ to G as shown in (5.36).
Recover $\mathbf{u}$ from $\mathbf{w}_i$ as $\mathbf{u} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \end{bmatrix} \mathbf{w}_i$, where $\boldsymbol{I}$ is the identity matrix.

5.2.4.3 Specific aggregation examples with goal regularization

Examples of the various ℓ_2-goal regularizers were presented in [99]. In this chapter, we extend these examples to include the ℓ_1-regularizers, the results of which are reported in Table 5.1. The following sections describe specific aggregation examples using both ℓ_1- and ℓ_2-goal regularizations. The value of the regularization parameter for ℓ_1-goal regularization

(t) is given in each of the following subsections, and unless explicitly stated otherwise $\lambda = 1$ for each ℓ_2-goal regularization experiment that follows[3].

Example 5. *Minimum aggregation* When we set the goal $\mathbf{g} = 0$, it reduces the regularization function in (5.3) to ℓ_p-norm regularization of the FM. The effect of this can be observed in Example 4 where the FM values of the untouched nodes were set to values close to the lowest end of the allowable range based on the monotonicity constraints. In the ℓ_p-min columns of Table 5.1, the values of untouched nodes are forced close to zero with our choice of ℓ_p-norm regularization. Comparing this with the aggregation operators discussed in Section 5.2.1, we notice that when the goal g is set to zero, the Choquet integral is forced to behave like a minimum operator. For the ℓ_1- regularized example $t = 3.23$.

Example 6. *Maximum aggregation* When we define the goal FM as all 1s, it results in the FM values of untouched nodes to default to the maximum end of their permissible range based on the monotonicity constraints. This essentially tunes the behaviour of the Choquet integral closer to the maximum aggregation (see Section 5.2.1). The effects of application of this goal on the example in Section 5.2.3 can be observed in Table 5.1. Here, in the columns corresponding to the ℓ_1-max and the ℓ_2-max goal, it is apparent that the untouched nodes are assigned the values closer to the maximum permitted based on the monotonicity constraints. In Table 5.1, we notice a perceptible effect of ℓ_2-goal regularization even on the data-supported nodes ($g(\{x_1\})$, $g(\{x_3\})$, $g(\{x_1, x_3\})$, and $g(\{x_2, x_3\})$), while they

[3]Experiments have shown that the behavior of ℓ_1-goal regularization is much more sensitive to t than that of ℓ_2-goal regularization to λ.

are pushed slightly farther from the underlying FM, the effect of regularization is milder compared to the untouched nodes. The degree to which ℓ_2-goal regularization pushes the FM values closer to the goal depends on the choice of λ; a larger value of λ forces the learned FM to look like the goal g, and in the process it may even force the data-supported nodes farther from the underlying values. As previously mentioned, for these experiments λ was arbitrarily set to 1 for ℓ_2-goal regularization; $t = 2$ for ℓ_1-goal regularization.

Example 7. *Mean aggregation* In this example, we define the goal of the FM to be that of mean aggregation as explained in Section 5.2.1. The learned FMs for this goal are presented in the columns labelled as ℓ_1-mean and ℓ_2-mean in Table 5.1. We observe that the FMs learned with the mean aggregation regularizers are more accurate than the FMs learned with the maximum aggregation regularizers. We attribute this improvement to the fact that the mean aggregation goal is more similar to the underlying FM than the maximum aggregation goal. In essence, the mean aggregation splits the difference between the extreme max and min aggregations. Finally, we note the results of both ℓ_1- and ℓ_2-goal regularized experiments are almost identical, where $t = 0.5$ for the ℓ_1-goal regularized experiment.

Remark 2. It is worth noting that while we are trying to address the data-unsupported FM variables by adding regularization to the QP as shown in (1.10), regularization will affect all FM variables. However, the effect will be most pronounced on the data-unsupported FM variables. Furthermore, if an FM variable is supported by numerous observations in

the training data then the SSE term in (1.10) will dominate and thus the regularization term will not contribute significantly to the learned value. This behavior is observed in Table 5.1 where we see approximately constant values learned for data-supported variables despite changing the regularization functions; the values learned for the data-unsupported variables vary widely based on the regularization function used.

5.2.5 Learning the Goal

The previous section described a strategy for learning an FM when knowledge of the underlying FM is known and can be encoded in the QP; however, information regarding the underlying FM is rarely available in real-world problems. This section presents a method for addressing this issue by simultaneously learning the FM at data-supported nodes and learning an appropriate goal for the remaining nodes. This method can also be used to enforce certain structure of the FM, which we demonstrate by restricting the number of degrees of freedom the FM takes. Specifically, we assume the underlying FM can be approximated by a *linear order statistic* (LOS)—to be precise, an *ordered weighted average* (OWA)[130]—which is a special case of the Choquet integral.

It is important to note that when we identify a much simpler model, like the OWA, as the goal structure for the FM, the algorithm does not strictly enforce the FM to take the form of an OWA. The degree to which the FM variables are pushed towards an OWA structure is

based on the trade-off between the regularization parameter, λ, and the variables' support in the training data. For a moderate λ value, when nodes are frequently visited by the training data, the SSE term in the objective function at (1.10) dominates and the regularization term has little effect. These nodes effectively learn the input-output relationship of the training data. However, the regularization term will dominate for nodes that are not frequently visited (i.e., not data-supported) and so these FM values will be more significantly pushed toward the goal FM. Thus, our goal-based approach learns the (potentially non-linear) aggregation relation through the parts of the FM that are data-supported, and defaults to the simpler OWA (or any other simpler goal) on the parts of the FM that lack support in the data. This approach combines the benefits of both the nonlinear FM aggregation as well as the simpler goal.

5.2.5.1 Defining an FM from an LOS

A normalized[4] LOS for a sample $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ is

$$L_{\mathbf{v}}(\mathbf{x}) = \sum_{i=1}^{m} v_i x_{(i)} = \mathbf{v}^T \mathbf{x}, \tag{5.12}$$

where $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(m)}$, $v_i \geq 0$, and $\sum_{i=1}^{m} v_i = 1$. Note that the ordering of the elements is similar to that of the Choquet integral; thus, it should come as no surprise that

[4]The generalized LOS is $\frac{\sum_{i=1}^{m} v_i x_{(i)}}{\sum_{i=1}^{m} v_i}$, but we adapt the constraint $\sum_{i=1}^{m} v_i = 1$ in our formulation, constraining the LOS to be an OWA.

the normalized LOS can be represented by a Choquet integral with respect to a *symmetric* FM[5].

Let the vector $\mathbf{v} \in \mathcal{R}^m$ now represent the LOS weight vector we wish to learn. We define the matrix $A \in \mathcal{R}^{(2^m-1)\times m}$ to map the LOS weight vector $\mathbf{v}$ into the domain of the FM $\mathbf{u}$. Then, for any $\mathbf{v}$ we can form an equivalent FM as $\hat{\mathbf{u}} = A\mathbf{v}$. For example, consider the LOS $\mathbf{v} \in \mathcal{R}^3$ and the FM vector has the ordering $\hat{\mathbf{u}} = (g(\{x_1\}),\ g(\{x_2\}),\ g(\{x_3\}),\ g(\{x_1, x_2\}),\ g(\{x_1, x_3\}),$ $g(\{x_2, x_3\}),\ g(\{x_1, x_2, x_3\}))$. Then A is defined as

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.13}$$

where each row sifts an LOS weight from $\mathbf{g}$ and assigns it to its corresponding FM term in $\hat{\mathbf{u}}$.

[5]An FM is symmetric if $\forall|A| = |B|,\ g(A) = g(B)$.

5.2.5.2 ℓ_2-LOS Regularization

Define the ℓ_2-LOS regularization function as

$$v_*(\mathbf{u}) = \lambda \|\mathbf{u} - A\mathbf{g}\|_2^2. \tag{5.14}$$

Similar to our regularizers presented in Section 5.2.4, this regularizer serves to push the solution of $\mathbf{u}$ towards the LOS represented by $\mathbf{g}$—however, here the matrix A is mapping the m-length LOS $\mathbf{g}$ to a symmetric FM $A\mathbf{g}$. Allowing $\mathbf{g}$ to be a learned variable (i.e., a learned LOS), this regularization function is minimized when

$$\mathbf{g} = (A^T A)^{-1} A^T \mathbf{u} = A^\dagger \mathbf{u}, \tag{5.15}$$

where $A^\dagger$ is the pseudo-inverse of A. Substituting (5.15) into (5.14) give the final form of the ℓ_2-LOS regularization function,

$$v_*(\mathbf{u}) = \lambda \|\mathbf{u} - AA^\dagger \mathbf{u}\|_2^2 = \lambda \|(\mathbf{I} - AA^\dagger)\mathbf{u}\|_2^2. \tag{5.16}$$

It can be shown that $(\mathbf{I} - AA^{\dagger})$ is a $(2^m - 1) \times (2^m - 1)$ block-diagonal matrix,

$$(\mathbf{I} - AA^{\dagger}) = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & B_m \end{bmatrix}, \tag{5.17}$$

where the ith block B_i is an $\binom{m}{i} \times \binom{m}{i}$ centering matrix,

$$B_i = \mathbf{I} - \frac{1}{\binom{m}{i}} \mathbf{1}\mathbf{1}^T. \tag{5.18}$$

Hence, we have a closed form solution for the block-diagonal matrix at (5.17); no inverses needed. The regularizer at (5.16) can simply be substituted into (1.10) and solved in the same manner.

Remark 3. The regularizer at (5.16) is a Tikhonov regularization, where the Tikhonov matrix is $\Gamma = (\mathbf{I} - AA^{\dagger})$ [121]. This regularizer can be further explained by examining how the block-diagonal form at (5.17) interacts with the FM $\mathbf{u}$. Essentially, each block of $(\mathbf{I} - AA^{\dagger})$ centers—i.e., removes the mean—of the respective components in $\mathbf{u}$. The components of $\mathbf{u}$ for each B_i are the FM values in the ith level of the lattice. That is, B_1 interacts with the densities, B_2 interacts with the duos, etc. Hence, this regularizer seeks FMs where the values at each level of the lattice are the same, i.e., symmetric FMs.

5.2.5.3 ℓ_1-LOS Regularization

Now consider the ℓ_1-LOS regularization function as

$$v_*(\mathbf{u}) = \lambda\|(\mathbf{I} - AA^\dagger)\mathbf{u}\|_1 = \lambda\|\Gamma\mathbf{u}\|_1, \tag{5.19}$$

where $\Gamma = \mathbf{I} - AA^\dagger$. Similar to Section 5.2.4.2, we move the regularization term to the constraints and use Tibshirani's iterative lasso algorithm. We first define a vector of signs of the function within the ℓ_1 norm as

$$\boldsymbol{\delta}_i = \text{sign}\left(\Gamma\mathbf{u}_i\right), \tag{5.20}$$

allowing us to calculate the ℓ_1 norm as $\boldsymbol{\delta}_i^T\Gamma\mathbf{u}_i$. Next we define the matrix including all $\boldsymbol{\delta}$s from iteration 0 as

$$G^i = \begin{bmatrix}\boldsymbol{\delta}_0^T & \boldsymbol{\delta}_1^T & \cdots & \boldsymbol{\delta}_i^T\end{bmatrix}^T \tag{5.21}$$

and denote its jth row as G_j^i. Using this notation, Algorithm 7 describes the process of solving the ℓ_1-LOS regularized QP.

Example 8. *LOS aggregation.* Applying ℓ_1- and ℓ_2-LOS regularization ($\lambda = t = 1$) to our example in Section 5.2.3 leads to the learned FMs in Table 5.1. Similar to the previous methods, the FM of the data-supported nodes is learned with high accuracy in both

Algorithm 7: ℓ_1-LOS Regularization via Tibshirani's Lasso Algorithm

Data: The QP at (3.7) and regularization parameter t.
Result: $\mathbf{u}$ - Lexicographically ordered FM vector.
$i = 0$;
$\hat{\mathbf{u}}_0 \leftarrow$ solve the unregularized QP in (3.7);
$G^0 \leftarrow$ calculate $\boldsymbol{\delta}_0$ and form the matrix in (5.21);
while $G^i_{i+1}\Gamma\hat{\mathbf{u}} > t$ **do**
  $i \leftarrow i + 1$;
  $\hat{\mathbf{u}}_i \leftarrow$ solve the QP in (3.7) with the additional constraint(s) $G^{i-1}\Gamma\hat{\mathbf{u}}_i \leq \mathbf{t}$;
  $G^i \leftarrow$ calculate $\boldsymbol{\delta}_i$ and append to G^{i-1} as shown in (5.21);
$\mathbf{u} \leftarrow \hat{\mathbf{u}}_i$.

cases. What is noteworthy, however, is how the nodes untouched by the training data are learned; the learned FM assigned to these untouched nodes is essentially the mean of the touched nodes at the same level in the lattice (touched nodes with the same cardinality). In other words, $g(\{x_2\}) = \frac{g(\{x_1\})+g(\{x_3\})}{2}$ and $g(\{x_1, x_2\}) = \frac{g(\{x_1,x_3\})+g(\{x_2,x_3\})}{2}$. Note that for this particular example, it is only a coincidence that the learned FM term $g(\{x_2\})$ exactly matches the underlying FM for ℓ_2-LOS regularization. The LOS-based regularization essentially encodes the knowledge that FM terms representing similar-sized sets should have approximately the same value.

5.2.6 Synthetic Experiments

The contrived examples in Table 5.1 showed that the underlying FM is accurately learned for data-supported terms in the FM. In this section, we design a synthetic data-based experiment to evaluate the ability of the learning algorithm to accurately learn the FM from training data. For each experiment, we randomly generated a *ground-truth*-FM (GT-FM)

using the node-wise algorithm presented in [62]. A synthetic 6-dimensional data set (six data columns of random real numbers generated from a normal distribution with zero-mean and unit standard deviation) was aggregated through this GT-FM to produce an output representing the target value. This output, when less than zero, is tagged as belonging to the -1-class, and the rest are tagged as the $+1$-class, which generates the ground-truth output labels. This 6-dimensional data set and corresponding ground-truth labels were used for training and testing a new FM. We trained the new FM using 80% of the total (1,000) observations, and evaluated its performance using the remaining 20%. Essentially, this experiment tests the ability of our learning algorithm to accurately learn the FM that originally produced the training data; hence, we expect very good performance (assuming our algorithm is good).

5.2.6.1 Learned FM performance

Since the path that an observation in the data set takes during the aggregation through the FM depends on its sort order, we have $6! = 720$ such unique paths for the 6-dimensional data. This number explodes quickly with additional dimensions in the data set. Thus, very frequently, we might not have an adequate number of observations in the data set to cover all possible paths in the FM lattice. In order to evaluate the ability to learn an FM in situations where the data are restricted to a small number of lattice paths, we artificially arranged our synthetic training data to take only a specific number of paths in our experiments (that

Table 5.2
Results of learning the fuzzy measure with synthetic data. The results presented in this table are the average F_1 scores from the 100 experiments described in Section 5.2.6.1.

	Number of Training Paths			
Algorithm	**1-path**	**1%**	**5%**	**10%**
None	0.847	0.936	0.991	0.996
ℓ_1**-min**‡	0.328	0.830	0.986	0.995
ℓ_2**-min**†	0.329	0.830	0.986	0.995
ℓ_1**-max**	0.847	0.937	0.991	0.996
ℓ_2**-max**	0.839	0.913	0.989	0.995
ℓ_1**-mean**	0.847	0.936	0.991	0.996
ℓ_2**-mean**	0.818	0.919	0.988	0.995
ℓ_1**-LOS**	0.847	0.937	0.990	0.996
ℓ_2**-LOS**	0.886	0.948	0.990	0.996

‡Equivalent to ℓ_1-regularization directly on the FM, i.e., $v_*(\mathbf{u}) = \lambda\|\mathbf{u}\|_1$.
†Equivalent to ℓ_2-regularization directly on the FM, i.e., $v_*(\mathbf{u}) = \lambda\|\mathbf{u}\|_2$.

is, a limited number of sort orders). Note, however, the testing data (200 observations) were perhaps more diverse than the training data since they were not constrained to follow the same paths as the training data; the testing data paths were chosen randomly.

One hundred trials of each experiment were run. The results presented are the classification performances measured using F_1 scores, defined in Section 5.2.6.2. The averages of the F_1 scores from the 100 trials are reported in Table 5.2; the columns in the table show the results of these experiments on 1-path, 1% of paths, 5% of paths, and 10% of paths, respectively.

5.2.6.2 $\mathbf{F}_1$ Score

The F_1 score is a measure of classification accuracy. It considers both the *Precision* and *Recall*, which are calculated using four parameters: *true positives* (TP), *false positives* (FP), *true negatives* (TN), and *false negatives* (FN). For example, an instance of a model prediction is considered as a TP when the predicted label as well as the actual label are both $+1$. Using these parameters, Precision and Recall are calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \tag{5.22}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{5.23}$$

The F_1 score is the harmonic mean of Precision and Recall,

$$F_1 \text{ score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}}. \tag{5.24}$$

5.2.6.3 Results

For the aggregation of 6-dimensional data, though we have $6! = 720$ unique sort orders, the results from our synthetic data experiments indicated that we can accurately learn a new FM by using training data comprising just 10% of the total possible sort orders. Even with 1% of paths (7 out of 720 sort orders), we were able to achieve F_1 scores greater than 0.90.

Figure 5.1 compares the GT-FM with the learned FM trained using just one sort order, which is analogous to 1-path through the FM lattice[6]. Here we note that the nodes lying on the training path are essentially perfectly learned, while many others have a visually perceptible mismatch with respect to the GT-FM. Regularization had no positive effect on cases where the FM was trained with at least 5% of the sort orders (36 out of 720). For the case with 1-path and the 1% of paths (7 out of 720), the ℓ_2-LOS regularization improved the F_1 score by 4.5% and 1.0%, respectively. Hence, this presents evidence that the LOS ℓ_2-LOS regularization is effective for imputing the untouched FM values when learning with insufficient training data.

5.2.7 Real-World Experiments

We ran experiments with real world data from the UCI Machine Learning repository [33] using our novel regularization functions summarized in Sections 5.2.4 and 5.2.5, as well as the unregularized objective at (3.7). We chose 14 classification data sets that cover a wide range of decision problems including the areas of health care, biology, material science, environmental science, and physics. Also, the selected data sets vary widely in the

[6]In Figure 5.1, each dark node corresponds to an FM value in the GT-FM, and the size of each node is scaled proportional to its value. The learned FM lattice was overlaid on top of this GT-FM lattice, and is marked in gray. So, for nodes where the learned FM-value was larger than the GT-FM value, the gray circle extends beyond the darker GT-FM node, representing the mismatch. When the learned FM-value is less than the GT-FM value, the dark node will have an inner thin circle marked by a white edge. When both the GT-FM value and the learned value exactly match, you only see the dark node. The line segments connecting the nodes represent the FM lattice path corresponding to the single sort order of the data used for training the FM.

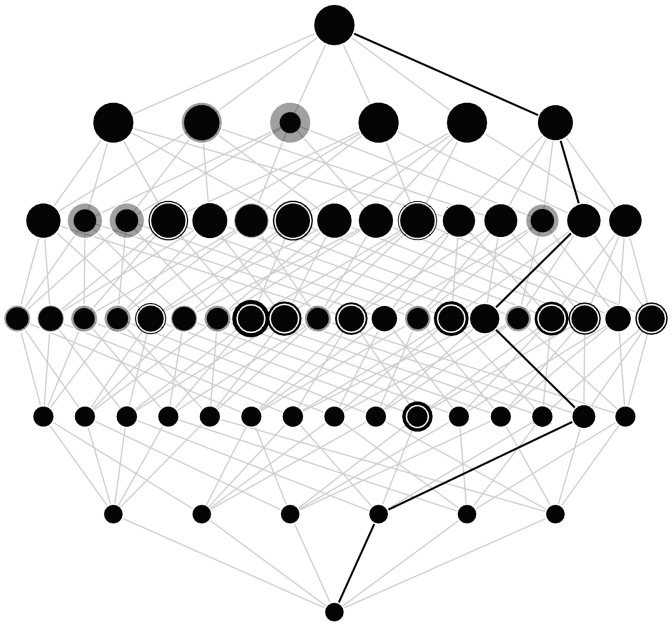

Figure 5.1: Learned FM where simulated data is restricted to just one path through the lattice[6] (no regularization). The gray lines represent all the possible paths (sort orders), the dark line represents the single path to which the simulated data was restricted.

number of features and the number of observations, adding to the diversity. The details on each of these data sets are summarized in Table A.2 in Appendix A. In addition, we also compared the performance of our Choquet integral-based aggregation method with several ensemble methods: AdaBoostM1 [39], LogitBoost [40], RUS-Boost [106], Subspace [11], Bagging [17], and MKLGL [126]. All the ensemble methods, except for MKLGL were implemented using the MATLAB's Statistics and Machine Learning Toolbox [84]. Note that MKLGL is not a decision-level fusion approach and is specific to multiple-kernel SVM learning; it learns a single unified kernel space by combining together base kernels. We include it, however, as it is one of the leading approaches for fusing multiple kernel SVMs. Experiments utilizing support vector machines, including the MKLGL experiments, used

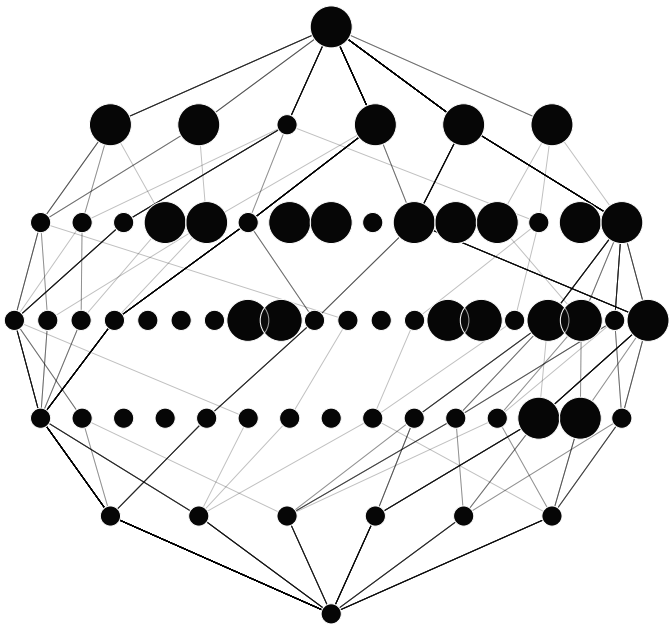

Figure 5.2: One example of a learned FM for the classification of the real-world breast cancer data set with no regularization. 48 out of 64 nodes (75%) were data-supported. As shown in Table 5.5 this algorithm achieved a mean F_1 score of 0.773 across the 100 trials.

the popular SVM library, LIBSVM [24]. 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 are sequestered for testing; the results we report comprise the mean and standard deviation of F_1 score on the testing data. Finally, for the regularized learning algorithms, we vary the regularization parameter, λ, to explore its effect on the F_1 score, and the results with the best λs are reported (i.e., a grid search). Essentially, we are comparing the best score from each algorithm.

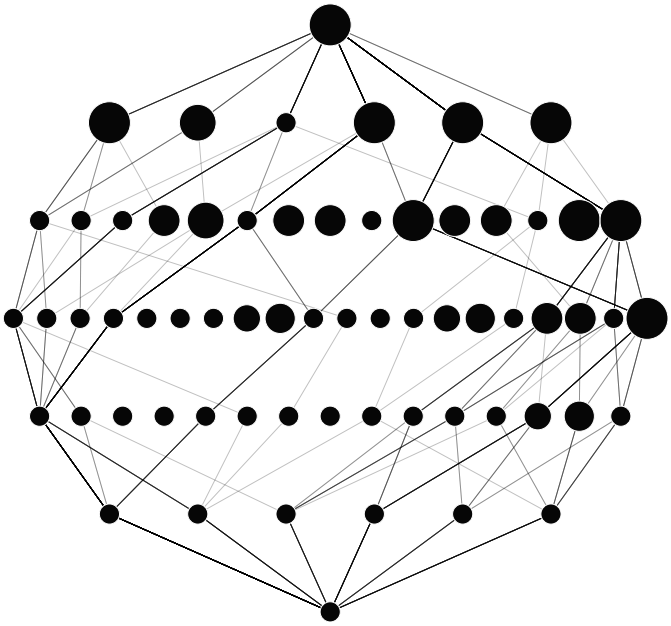

Figure 5.3: One example of a learned FM for the classification of the real-world breast cancer data set after applying ℓ_1-mean regularization, which increased the mean F_1 score to 0.779. With this regularization there is much less variability in the learned FM values across each level in the lattice.

5.2.7.1 Results

Table 5.5, available in Appendix 5.5, summarizes the results of these experiments. The results in Table 5.5 were used to detect statistically significant differences among methods. Specifically, we performed an $n \times n$ pairwise comparison of our proposed methods and the other state-of-the-art algorithms using the non-parametric statistical procedure recommended by Garcia et al. [42]. The p-values calculated for the comparison of each pair of methods are adjusted using the Nemenyi (NM)[96], Holm (HM) [63], and the Shaffer (SH)[107] methods. Table 6.2 summarizes the results of this comparison where we see that by using the unadjusted p-value, both mean regularizations (ℓ_1 and ℓ_2) are considered best,

Table 5.3
Comparison of our proposed methods and other state-of-the-art algorithms on real-world data using non-parametric evaluation methods.

# of instances of best performance based on each evaluation method				
	Non-parametric methods			
$v_*(\mathbf{u})$	Unadjusted†	NM [96]	HM [63]	SH [107]
None	3	0	0	1
ℓ_1**-min**	3	1	1	1
ℓ_2**-min**	5	1	1	2
ℓ_1**-max**	10	**3**	**3**	**3**
ℓ_2**-max**	7	**3**	**3**	**3**
ℓ_1**-mean**	**11**	**3**	**3**	**3**
ℓ_2**-mean**	**11**	**3**	**3**	**3**
ℓ_1**-LOS**	3	0	0	1
ℓ_2**-LOS**	7	**3**	**3**	**3**
Simple-min	0	0	0	0
Simple-max	2	0	0	0
Simple-mean	3	1	1	1
AdaBoostM1	1	0	0	0
LogitBoost	1	0	0	0
GentleBoost	1	0	0	0
RUS-Boost	0	0	0	0
Subspace	0	0	0	0
Bag	1	0	0	0
MKLGL	7	**3**	**3**	**3**

†Non-parametric p-value calculated as per Garcia et al. [42]. We considered an initial α = 0.05 and adjusted it using the Nemenyi (NM)[96], Holm (HM) [63], and the Shaffer (SH)[107] methods.

each with 11 instances of superior performance.

The three adjusted p-value approaches tell a slightly different, but consistent story. Since we are comparing $\binom{19}{2}$ = 171 pairs, the adjusted p-values are quite conservative, and the best performing mean regularizations are now considered to have only 3 instances of superior performance for each adjustment method (NM, HM, and SH). Furthermore, other regularizations and algorithms are also achieving "equivalent" results when analyzed using these adjusted p-value methods. The adjusted p-value analysis shows that the mean, max, and ℓ_2-LOS regularizations, as well as the MKLGL algorithm all achieve superior results.

5.3 Learning the ChI in the Presence of Uncertainty Caused by Limited Training Data Volume and Variety

5.3.1 Introduction

The regularization-based approach for training the ChI presented in this chapter addresses the issue of limited training data support by applying a form of Tikohonv regularization that regularizes the learned model to a prototype model or model-type, e.g., an OWA operator. The regularization implicitly sets the values for data-unsupported variables, thus eliminating the bias of initialization. However, this goal-based regularization approach uniformly regularizes all the FM values of ChI, irrespective of the corresponding data-support.

In this section, we incorporate the *data support of a variable* and the *degree of disparity* between sources (the relative support across all variables) to develop a method that produces good FM solutions that explicitly consider the uncertainty due to limited training data volume and variety during the training process. In this approach, the amount of regularization is directly related to the degree of support, i.e., variables with a strong data-support will be regularized to a very low degree, while variables with limited support will be regularized

to a greater extent.

5.3.2 Methodology

The full form of QP to learn the FM $\mathbf{u}$ for fusion of m classifiers is

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T\hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u}, \quad C\mathbf{u} \leq \mathbf{0}, \quad (\mathbf{0}, 1)^T \leq \mathbf{u} \leq \mathbf{1}, \tag{5.25}$$

where C is the $(m(2^{m-1}-1)) \times (2^m - 1)$ constraint matrix of $\{0, 1, -1\}$ values,

$$C = \begin{bmatrix} 1 & 0 & \cdots & -1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & -1 \end{bmatrix}. \tag{5.26}$$

Adding the regularization term proposed (5.3) to (5.25) gives

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T\hat{D}\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda\|\mathbf{u} - \mathbf{g}\|_p, \quad C\mathbf{u} \leq \mathbf{0}, \quad (\mathbf{0}, 1)^T \leq \mathbf{u} \leq \mathbf{1}, \tag{5.27}$$

We propose a regularization approach that explicitly takes into consideration the data support during the training process. Let the degree of support for variables $\mathbf{u}$ be $\mathbf{d} =$

$(d_{u_1}, d_{u_2}, \ldots, d_{u_X})$, where each element of $\mathbf{d}$ is a quantification of the data support for respective FM variable. We express $\mathbf{d}$ as support measure relative to the *expected ideal data support*. Each layer of the FM has $\binom{m}{l}$ nodes, where l is the number of members in the set represented by the FM nodes. For example, $l = 2$ for $\mu(\{x_1, x_2\})$. The data support vector $\mathbf{d}$ is thus calculated as the number of training data visits to an FM node during training, which we denote as k_i, divided by the expected frequency of data visits, $\frac{1}{\binom{m}{l}}$. In the ChI at (1.4), this calculation is $\frac{1}{\binom{m}{|S|}}$.

Consider an example where there are $m = 4$ sources, with $K = 100$ training data examples. The ideal data support for each FM node that represents the singletons—e.g., $\mu(\{x_1\})$—would be $\frac{100}{\binom{4}{1}} = 25$ training examples. The FM nodes that represent the duets of sources—e.g., $\mu(\{x_1, x_2\})$—should be supported by $\frac{100}{\binom{4}{2}} \approx 17$ training examples. Thus, FM nodes that are supported more than expected have a higher than expected data support, and vice versa for those that are supported less. The form of the data support vector is thus

$$d_i = k_i \binom{m}{|S_i|}, \quad i = 1, \ldots, 2^m - 1, \tag{5.28}$$

where k_i is the number of times the training data support the ith FM variable, out of a possible K training examples, and $|S_i|$ is the cardinality of the set of sources that the FM variable represents.

Given a data support vector $\mathbf{d}$, we can now apply regularization to the FM learning accordingly. Let the optimization at (5.4) be extended as

$$\min_{\mathbf{u}} \mathbf{u}^T D\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda||\mathbf{e_d} \circ (\mathbf{u} - \mathbf{g})||_p, \tag{5.29}$$

where $\mathbf{e_d} = (e^{-\sigma d_{u_1}/2}, e^{-\sigma d_{u_2}/2}, \ldots, e^{-\sigma d_{u_X}/2})^T$, σ attenuates the effect of the data support regularization, and $\mathbf{g}$ is an FM goal, e.g., a min, max, or mean FM. Large σ will result in less regularization on highly supported FM variables (i.e., large values of $\mathbf{d}$), while small σ allows more regularization on supported FM variables.

5.3.3 ℓ_2 regularizaiton

For $p = 2$, the regularizer in (5.29) can be reduced to

$$\begin{aligned}\lambda||\mathbf{e_d} \circ (\mathbf{u} - \mathbf{g})||_2 =& \lambda\big[\mathbf{u}^T\text{diag}(\mathbf{e_d^2})\mathbf{u} - 2\mathbf{g}^T\text{diag}(\mathbf{e_d^2})\mathbf{u} + \\ & \mathbf{g}^T\text{diag}(\mathbf{e_d^2})\mathbf{g}\big],\end{aligned} \tag{5.30}$$

where $\text{diag}(\mathbf{e_d^2}) = \text{diag}(\mathbf{e_d} \circ \mathbf{e_d})$ and is the diagonal matrix,

$$\text{diag}(\mathbf{e_d^2}) = \begin{bmatrix} e^{-\sigma d_{u_1}} & 0 & \cdots & 0 \\ 0 & e^{-\sigma d_{u_2}} & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & e^{-\sigma d_{u_X}} \end{bmatrix}.$$

Hence, it is easy to see that this new optimization can be solved by a QP with new Hessian and linear terms,

$$\hat{D} = D + \lambda \text{diag}(\mathbf{e_d^2}), \quad \hat{\mathbf{f}} = \mathbf{f} - 2\lambda \mathbf{g}^T \text{diag}(\mathbf{e_d^2}). \tag{5.31}$$

This regularizer is equivalent to an equal-error elliptical regularizer surface centered at $\mathbf{g}$ with axes lengths determined by the data support.

5.3.4 ℓ_1 regularization

The optimization at (5.29) with $p = 1$ yields

$$\min_{\mathbf{u}} 0.5\mathbf{u}^T D\mathbf{u} + \mathbf{f}^T\mathbf{u} + \lambda \sum_i |(\mathbf{e_d})_i (u_i - g_i)| \,; \tag{5.32}$$

however, this regularizer cannot be reduced or combined into the objective function as was done with ℓ_2-regularization. This is because the difference term within the absolute value will not always be non-negative. To address this, we move the regularization term to the constraints through the use of Tibshirani's iterative lasso algorithm; see [119] for a brief description.

This algorithm solves the unregularized problem while iteratively updating the constraints to enforce sparsity in the difference term $(\mathbf{e_d} \circ (\mathbf{u} - \mathbf{g}))$; although, the problem in (5.32) must be reformulated as follows. Let us first lump $(\mathbf{e_d} \circ \mathbf{u})$ and $(\mathbf{e_d} \circ \mathbf{g})$ into a single long vector $\mathbf{w}$ as $\mathbf{w} = [(\mathbf{e_d} \circ \mathbf{u})^T \quad (\mathbf{e_d} \circ \mathbf{g})^T]^T \in \mathcal{R}^{2^{m+1}-2}$. The unregularized QP at (5.25) can then be rewritten as

$$\min_{\mathbf{w}} 0.5\mathbf{w}^T \hat{D}_{\mathbf{w}} \mathbf{w} + \mathbf{f}_{\mathbf{w}}^T \mathbf{w}, \quad C_{\mathbf{w}} \mathbf{w} \leq \mathbf{0}, \quad \mathbf{b}_l \leq \mathbf{w} \leq \mathbf{b}_r, \tag{5.33}$$

where

$$\hat{D}_{\mathbf{w}} = \begin{bmatrix} \hat{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{f}_{\mathbf{w}} = [\mathbf{f}^T \ \mathbf{0}^T]^T, \quad C_{\mathbf{w}} = [C \ \mathbf{0}], \tag{5.34}$$

$$\mathbf{b}_l = [(\mathbf{0}, 1)^T \ \mathbf{g}^T]^T, \quad \mathbf{b}_r = [\mathbf{1} \ \mathbf{g}^T]^T.$$

Denote the vector of the sign of the differences at the ith iteration as

$$\boldsymbol{\delta_i} = \text{sign}(\mathbf{u}_i - \mathbf{g}), \tag{5.35}$$

and the matrix including all $\boldsymbol{\delta}$s from iteration 0 as

$$G^i = \begin{bmatrix} \boldsymbol{\delta_0}^T & -\boldsymbol{\delta_0}^T \\ \boldsymbol{\delta_1}^T & -\boldsymbol{\delta_1}^T \\ \vdots & \vdots \\ \boldsymbol{\delta_i}^T & -\boldsymbol{\delta_i}^T \end{bmatrix}, \tag{5.36}$$

such that multiplication of G^i_j, the jth row of G^i, with the vector $\mathbf{w}$ represents an ℓ_1-summation, e.g.,

$$G^i_j \mathbf{w}_i = \|\mathbf{e_{d_s}} \circ (\mathbf{u}_i - \mathbf{g})\|_1, \tag{5.37}$$

where $\mathbf{u}_i$ is the learned FM from the ith iteration of the algorithm. Note the regularization parameter is $t \propto 1/\lambda$, and the vectorized version is denoted as $\mathbf{t} = t\mathbf{1}$. See [68] for a detailed description of the iterative lasso algorithm for this problem, which follows the same notation presented in (5.35)–(5.37). The only change from [68] is the calculation of $G^i_j \mathbf{w}_i$ which here includes the application of support-based attenuation vector $\mathbf{e_d}$.

5.3.5 Types of FM goals

The vector $\mathbf{g}$ in (5.29) and rest of the Section 5.3.2 represents a goal of what we expect the underlying FM to look like for a particular aggregation problem, while the regularization parameters λ and σ define the degree to which the regularization is enforced by the training

algorithm. When $\mathbf{g} = 0$, the regularization function in (5.29) is reduced to ℓ_p-norm regularization of the FM, which pushes the FM towards a minimum aggregation operator, i.e., ℓ_p-min regularization. Similarly, when we define the goal $\mathbf{g}$ as all 1s, it tunes the behaviour of the Choquet integral to the maximum aggregation, i.e., ℓ_p-max regularization. When we define $\mathbf{g}$ as $g(A_i) = |A_i|/n, \forall A_i \subset X$, it acts as ℓ_p-mean regularization. See [68] for more details on types of regularization goals for FM.

5.3.6 Degree of disparity

To provide a quantitative measure of how well a training data set supports the learning of an FM, we propose a simple method to compute *degree of disparity*. If a training data set is equally distributed across the FM lattice; i.e., each path from $\mu(\emptyset)$ to (X) is traversed an equal number of times, then the expected visits to each node is $\frac{K}{\binom{m}{l}}$, where K is the number of training examples, m is the number of sources, and l is the cardinality of the set that the FM node represents, i.e., $l = |S_i|$. Hence, we can use the data support vector to directly compute a degree of disparity. Using $\mathbf{d}$, calculated by (5.28), we normalize to

$$\tilde{d}_i = \frac{d_i}{\|d\|_1},$$

The vector $\tilde{d}$ now represents a distribution of the training examples across the FM with respect to the ideal expected data support.

We then calculate the degree of disparity ζ as the entropy of this normalized support

$$\zeta = -\frac{1}{\log(2^m - 1)} \sum_{i=1}^{2^m-1} \tilde{d}_i \log(\tilde{d}_i), \tag{5.38}$$

where m is the number of sources.

5.3.7 Layer-level degree of disparity

While (5.38) provides us with a single degree of disparity for the entire FM lattice, we can calculate layer-level degree-of-disparity ζ_{layer} for individual FM lattice layers. ζ_{layer} provides us with a higher-resolution view of disparity of the data. We know that each training example can be represented as a path from $\mu(\emptyset)$ to $\mu(X)$; thus, all K training examples have to pass through each of the FM layers. As described in Subsection 5.3.6, for a uniformly distributed training data set, we expect $\frac{K}{\binom{m}{l}}$ visits to each node where m is the number of sources, and l is the cardinality of the set that the FM node represents, i.e., $l = |S_i|$. We calculate the layer-level degree of disparity for a layer j as

$$\zeta_{layer_j} = -\frac{1}{\log \binom{m}{l}} \sum_{i=1}^{\binom{m}{l}} \frac{k_{j_i}}{M} \log \left(\frac{k_{j_i}}{M} \right), \tag{5.39}$$

where k_{j_i} is the number of times the training data support the ith FM variable in layer j, out of a possible K training examples.

5.3.8 Visualization of data support

A convenient method to visualize an FM is to represent it as a lattice (i.e., Hasse diagram). In [100] we presented an approach to visualize the FM and the aggregation process of ChI. With this method, the aggregation of each permutation of source inputs can be represented as a path through the FM lattice.

Herein, we extend this method to visually present and evaluate the degree of disparity. Fig. 5.4 shows the visualization of the data-support on the FM lattice trained on the Vertebral [33] data set. The *layer entropy* shown on the left of the lattice is calculated using (5.39), and the *net entropy* is calculated using (5.38). Notice that only a small number of possible paths through the lattice were traversed, resulting in low entropy values, indicating a relatively low-support training data set.

5.3.9 Experiments

We evaluated our support-based regularization methods and the visualization approach summarized in section 5.3.2 on real world data sets from the UCI Machine Learning repository [33]. We chose 12 classification data sets that cover a wide range of decision problems. These data sets include the areas of health care, biology, material science, environmental science, and physics. Also, the selected data sets vary widely in the number of features

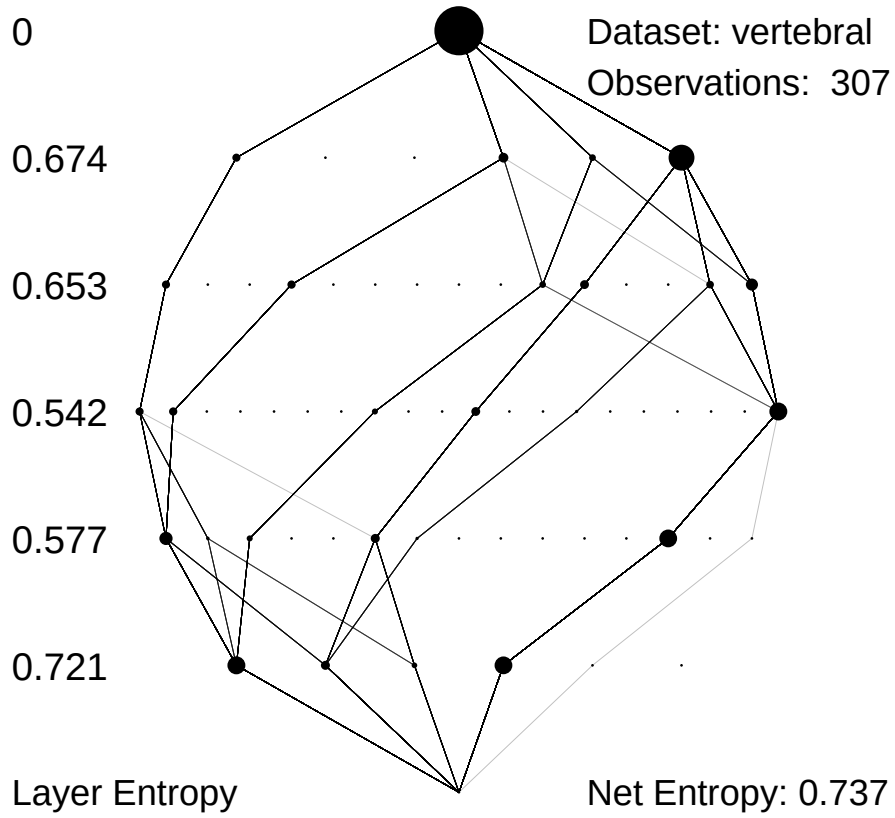

Figure 5.4: Visualization of data-support of FM lattice trained on Vertebral data set. Thickness of the lines is proportional to the frequency of data traversal, and node sizes are scaled proportional to the data-support.

and the number of observations, adding to the diversity. Each experiment consists of 10 random cross-fold trials, where in each trial a random partition of 80% of the data is used for training and the remaining data are used for testing. The classification performance was measured using the F_1 score. F_1 score is the harmonic mean of $precision$ and $recall$. See [109] for more details on F_1 score. The results we report comprise the mean and standard deviation of F_1 score on the testing data. For the regularized learning algorithms, we vary both the regularization parameter, λ, and the data-support-attenuation parameter σ to explore their effect on the F_1 score, and the best results with corresponding λ and σ are

reported based on a grid search.

For each data set, we calculated the degree of disparity presented in sections 5.3.6 and 5.3.7, and visualized the diversity of the data according to the approach presented in section 5.3.8. We analyzed the performance gains from our proposed regularization methods in correspondence with the data diversity visualizations.

5.3.9.1 Results

Table 5.4 summarizes the results of our experimental analysis to compare the performance of our regularization methods. We performed two-tailed t-tests to identify the regularization methods that outperformed the non-regularized algorithms at a 5% significance level, and such better performers were marked in bold in Table 5.4. For 10 out of 12 data sets, at least one our new regularization methods have significantly improved the performance. Specifically, $\ell_1 - max$ and $\ell_2 - max$ are the best performers with 6 best instances each. Also, among all the best instances (marked bold), 50% (15 out of 30) had a non-zero data-support-attenuation parameter σ, indicating a substantial supportive role played by our novel data-support-based attenuation approach in the performance of the regularization methods.

We visualized the degree of data disparity of the data sets to assess its relation with the performance gains achieved through our regularization methods. For data sets like Breast

Cancer and Heart, where the regularization has resulted in remarkable gains, we noticed the degree of data disparity to be relatively low. This is evident in Fig. 5.5, where only a small proportion of the possible data paths were traversed by the data samples, and the layer-level entropies were also relatively low. On the other hand, for data sets like Ionosphere and Ecoli, where none of the regularized methods have yielded any significant performance gain, we noticed a high degree of disparity, which can be observed in Fig. 5.6, where the layer-level entropy values are higher and data path diversity is richer relative to the Breast Cancer data set in Fig. 5.5. The results in Table 5.4, and the data-support visualizations of data sets reinforce our hypothesis that introducing a data-support-based sensitivity to our goal-based regularization methods improves the performance, particularly on the data sets with a lower degree of data-disparity.

5.3.9.2 Comparison with deep neural networks

To compare the performance of ChI-based decision fusion methods with DNNs, we built 3-hidden layer DNNs on the UCI classification data sets. Each hidden layer comprised 256 nodes. We performed 10 experimental trials—in each trial, a random permutation of the data set is split into 75%/25% for training and testing, respectively. The DNNs were trained for 40 epochs in each experiment. The performance of DNNs is presented in Table 5.4. Only on 3 out of the 10 data sets, the DNN model has either matched or outperformed the ChI aggregation methods, indicating a poor performance of DNNs. This is possibly

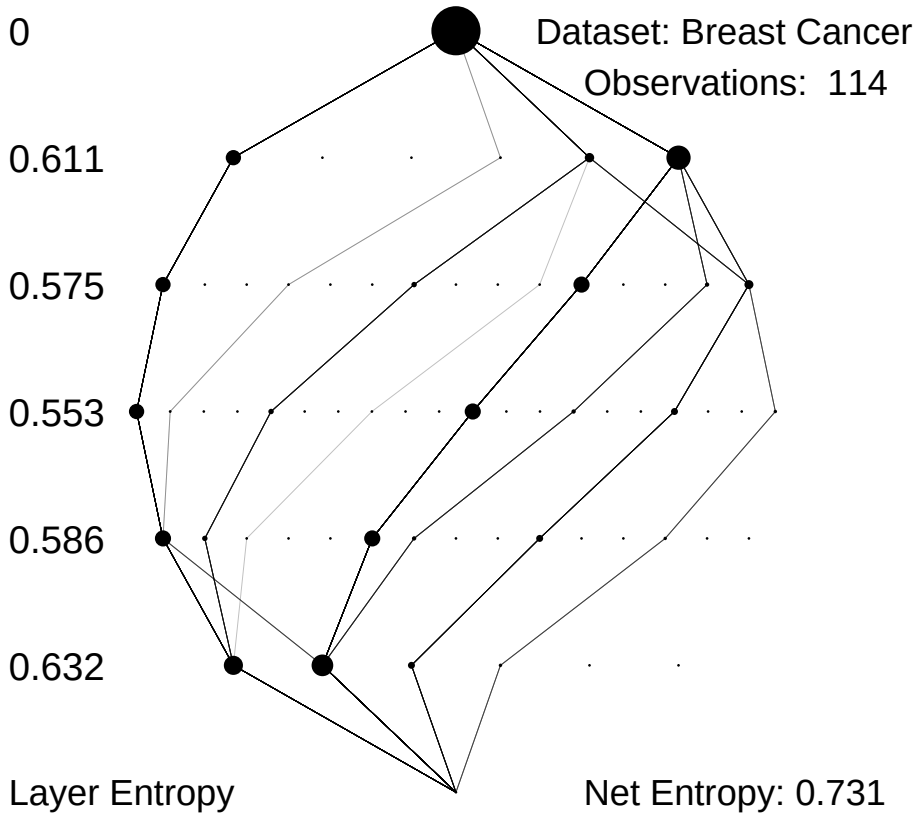

Figure 5.5: Support visualization for the Breast Cancer data set.

because the training data sets are not compatible with relatively large DNNs comprising more than 100,000 trainable parameters, resulting in potentially over-fit models. For the three data sets where the DNNs outperformed, the improved performance comes at a cost of loss of explainability of the learned solutions, which is readily available with the ChI models. The choice of ChI vs. DNN models depends on the users' preferences and the explainability requirements.

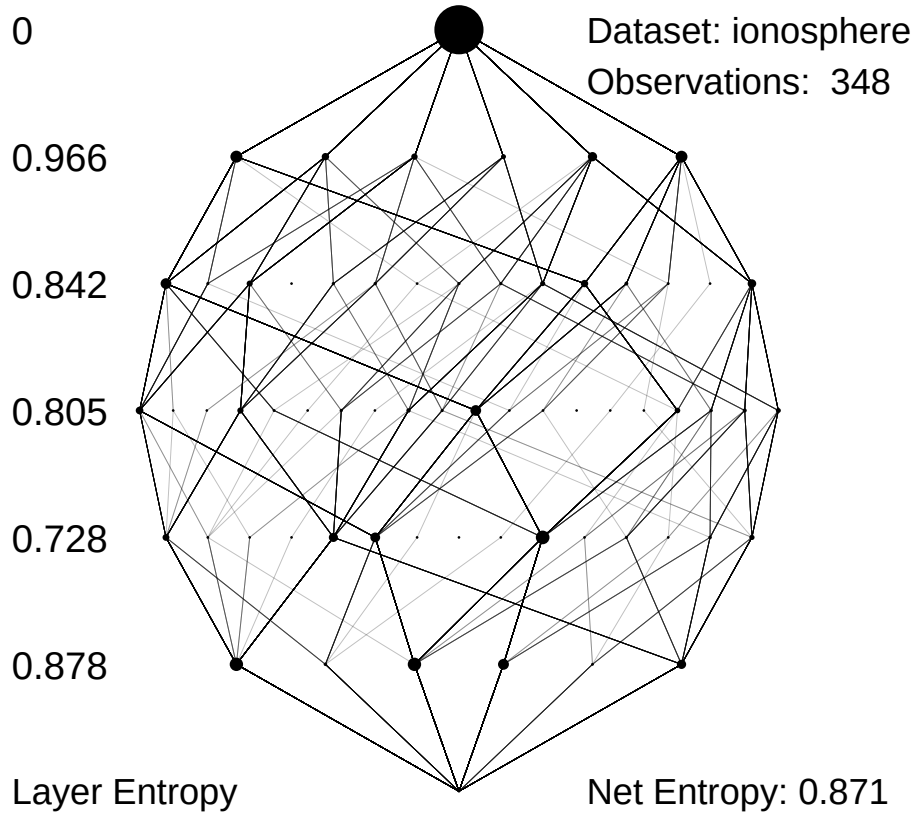

Figure 5.6: Support visualization for the Ionosphere data set. This data set has a higher degree of disparity, evident in the richer diversity of the paths traversed by the data samples.

Table 5.4

Performance of our proposed method on real-world data.

	Data Set [net entropy*]												
$v_*(\mathbf{u})$	Vertebral [0.737]	Biodeg [0.918]	Mmass [0.873]	Breast-Cancer [0.731]	Trans-fusion [0.924]	Heart [0.766]	Sonar [0.720]	Derma-tology [0.834]	Ecoli [0.646]	Glass [0.809]	Iono-sphere [0.871]	Absentee [0.760]	# of Best in-stances **(bold)**
None	0.542 (0.079)	0.731 (0.041)	0.791 (0.032)	0.051 (0.114)	0.437 (0.061)	0.116 (0.176)	0.763 (0.098)	0.903 (0.032)	0.837 (0.110)	0.889 (0.064)	0.942 (0.025)	0.227 (0.183)	0
ℓ_1**-min**	0.542 (0.079) λ=0.0 σ=0.0	0.731 (0.041) λ=0.0 σ=0.0	0.812 (0.031) λ=10 σ=5.0	**0.776 (0.114)** λ=100 σ=0.0	0.437 (0.061) λ=0.0 σ=0.0	**0.795 (0.047)** λ=100 σ=0.0	**0.862 (0.086)** λ=3 σ=0.01	0.903 (0.032) λ=0.0 σ=0.0	0.837 (0.110) λ=0.0 σ=0.0	0.889 (0.064) λ=0.0 σ=0.0	0.942 (0.025) λ=0.0 σ=0.0	**0.727 (0.043)** λ=100 σ=0.01	4
ℓ_2**-min**	0.542 (0.079) λ=0.0 σ=0.0	0.731 (0.041) λ=0.0 σ=0.0	0.813 (0.032) λ=5 σ=0.5	**0.776 (0.114)** λ=1000 σ=0.0	0.437 (0.061) λ=0.0 σ=0.0	**0.795 (0.048)** λ=10 σ=0.0	**0.862 (0.086)** λ=10 σ=0.1	0.903 (0.032) λ=0.0 σ=0.0	0.837 (0.110) λ=0.0 σ=0.0	0.889 (0.064) λ=0.0 σ=0.0	0.944 (0.025) λ=0.1 σ=0.1	**0.737 (0.038)** λ=100 σ=0.01	4
ℓ_1**-max**	**0.839 (0.056)** λ=10.0 σ=0.0	**0.853 (0.041)** λ=5.0 σ=0.0	**0.822 (0.028)** λ=0.5 σ=0.0	0.051 (0.114) λ=0.0 σ=0.0	**0.505 (0.056)** λ=100.0 σ=0.0	0.116 (0.176) λ=0.0 σ=0.0	0.763 (0.098) λ=0.0 σ=0.0	**0.976 (0.012)** λ=1.0 σ=0.0	0.912 (0.072) λ=100.0 σ=0.0	**0.937 (0.036)** λ=3.0 σ=0.0	0.942 (0.025) λ=0.0 σ=0.0	0.227 (0.183) λ=0.0 σ=0.0	6
ℓ_2**-max**	**0.839 (0.062)** λ=100.0 σ=0.01	**0.849 (0.039)** λ=1000.0 σ=0.001	**0.822 (0.030)** λ=1000 σ=0.5	0.051 (0.114) λ=0.0 σ=0.0	**0.505 (0.056)** λ=100.0 σ=0.0	0.116 (0.176) λ=0.0 σ=0.0	0.763 (0.098) λ=0.0 σ=0.0	**0.977 (0.013)** λ=1000.0 σ=0.5	0.912 (0.072) λ=100.0 σ=0.0	**0.937 (0.036)** λ=5.0 σ=0.0	0.947 (0.031) λ=0.5 σ=0.1	0.227 (0.183) λ=0.0 σ=0.0	6
ℓ_1**-mean**	0.542 (0.079) λ=0.0 σ=0.0	0.731 (0.041) λ=0.0 σ=0.0	**0.824 (0.028)** λ=0.5 σ=0.5	**0.582 (0.215)** λ=3 σ=0.5	0.450 (0.063) λ=0.1 σ=0.01	**0.767 (0.066)** λ=3.0 σ=3.0	0.821 (0.094) λ=0.5 σ=0.1	**0.972 (0.017)** λ=3.0 σ=0.1	0.837 (0.110) λ=0.0 σ=0.0	0.889 (0.064) λ=0.0 σ=0.0	0.955 (0.023) λ=1.0 σ=3.0	**0.739 (0.036)** λ=5.0 σ=0.1	5
ℓ_2**-mean**	0.542 (0.079) λ=0.0 σ=0.0	0.731 (0.041) λ=0.0 σ=0.0	**0.824 (0.030)** λ=100.0 σ=0.5	**0.556 (0.213)** λ=100 σ=0.0	0.437 (0.061) λ=0.0 σ=0.0	**0.764 (0.076)** λ=1000.0 σ=0.0	0.821 (0.095) λ=3.0 σ=0.01	**0.968 (0.018)** λ=100.0 σ=0.01	0.837 (0.110) λ=0.0 σ=0.0	0.889 (0.064) λ=0.0 σ=0.0	0.951 (0.032) λ=0.5 σ=0.0	**0.734 (0.032)** λ=1000.0 σ=0.0	5
Deep_learning** (132,864 parameters)	0.690 (0.004)	0.641 (0.043)	0.768 (0.019)	0.545 (0.110)	***0.786 (0.024)***	0.538 (0.015)	0.757 (0.077)	0.765 (0.046)	***0.924 (0.021)***	0.851 (0.032)	***0.988 (0.001)***	0.562 (0.043)	3

Each cell presents the F1-score—mean (std-dev), the corresponding best regularization parameter λ (when relevant), and the best support-based-attenuation parameter σ (when relevant) based on 10 experimental trials. Cells marked in bold (non-italicized) indicate better performance than non-regularized algorithm based on a two-tailed t-test at a 5% significance level. *Net entropy is calculated as per (5.38). **Deep learning model is comprised of three hidden layers with 256 nodes each. The cells marked in bold-italic are the data instances on which deep learning outperformed ChI-based methods.

5.4 Conclusions and Future Work

This chapter introduced new regularization functions that enable the user to encode knowledge of the underlying FM structure into the training algorithm's of ChI. Using these regularizers, the user can define a particular goal for the FM before learning. An alternative interpretation is that the regularization function can enforce a constraint on the structure of the learned FM (e.g., min-like, mean-like, or LOS-like). We discussed the application of the new regularization functions and demonstrated their behavior using synthetic and real-world data sets. Experimental results indicated that while there is (not surprisingly) no one-size-fits-all algorithm, the addition of the various goal regularizers proposed in this chapter enables state-of-the-art performance. This conclusion is supported by the statistical comparison based on the non-parametric methods. We also proposed a data support-based regularization strategy that selectively regularizes the learning FM variables according to their data support. The visualization approach presented in this chapter illustrate the training data support, and the corresponding quantitative entropy measures to calculate data disparity. The experimental evaluation of our new regularization strategy on real-world data sets demonstrated a meaningful complementary role played by our novel data-support-based attenuation method. Our method to visualize the training data support has further bolstered our hypothesis that data support-based regularization has a particularly strong impact on data sets with a low degree of data disparity.

Future work will focus on online/real-time optimization of the ChI, develop strategies to generate interval-valued FMs based on the uncertainty of the learned FMs, and provide upper and lower boundaries of the aggregation output. Another potential avenue for future research is the application of explainability indices [98], such as *Shapley* and *Interaction*, to explain different aspects of fusion while using interval-valued FMs under uncertainty.

5.5 Appendix

5.5.1 Tibshirani's Lasso Algorithm

Tibshirani proposed an iterative method of solving the ℓ_1-regularization problem in his seminal lasso regression work [120]. That method is summarized here for the case of a general objective function with ℓ_1-regularization, or

$$\min_{\mathbf{x}} J(\mathbf{x}) + \lambda \|\mathbf{x}\|_1, \tag{5.40}$$

where $\mathbf{x} \in \mathcal{R}^N$. The minimization at (5.40) can be rewritten as

$$\min_{\mathbf{x}} J(\mathbf{x}), \quad \text{s.t.} \quad \|\mathbf{x}\|_1 \leq t, \tag{5.41}$$

where $t \propto 1/\lambda$. This can be equivalently stated as

$$\min_{\mathbf{x}} J(\mathbf{x}), \quad \text{s.t.} \quad G\mathbf{x} \leq \mathbf{t}, \tag{5.42}$$

where $G \in \mathcal{R}^{2^N \times N}$ and the term $G\mathbf{x}$ represents all possible linear combinations of $\mathbf{x}$ with unit coefficients. For example, if $N = 2$ then

$$G = \begin{bmatrix} +1 & +1 \\ +1 & -1 \\ -1 & +1 \\ -1 & -1 \end{bmatrix}, \tag{5.43}$$

and since the number of rows of G grows as 2^N, the number of constraints quickly becomes intractable. To address this, an iterative algorithm is applied where constraints are added sequentially.

To solve (5.42), first solve

$$\hat{\mathbf{x}}_0 = \min_{\mathbf{x}} J(\mathbf{x}), \tag{5.44}$$

with $\boldsymbol{\delta}_i = \text{sign}(\hat{\mathbf{x}}_i)$ and

$$G_i = \begin{bmatrix} \boldsymbol{\delta}_0^T & \boldsymbol{\delta}_1^T & \cdots & \boldsymbol{\delta}_i^T \end{bmatrix}^T.$$

Then solve the constrained/regularized problem

$$\hat{\mathbf{x}}_i = \min_{\mathbf{x}} J(\mathbf{x}), \quad \text{s.t.} \quad G_{i-1}\hat{\mathbf{x}}_{i-1} \leq t, \tag{5.45}$$

until $\boldsymbol{\delta}_i^T \hat{\mathbf{x}}_i \leq t$.

Table 5.5
Performance of our proposed methods and other state-of-the-art algorithms on real-world data.

	Data Set													
$v_*(\mathbf{u})$	Vertebral	Biodeg	Mmass	Breast-Cancer	Transfusion	Heart	Sonar	Dermatology	Ecoli	Glass	Wine	Ionosphere	Absentee	Abalone
None	0.542 (0.114)	0.71 (0.043)	0.791 (0.032)	0.733 (0.013)	0.384 (0.068)	0.724 (0.029)	0.845 (0.039)	0.866 (0.054)	0.788 (0.14)	0.834 (0.096)	0.994 (0.009)	0.947 (0.021)	0.696 (0.016)	0.798 (0.007)
ℓ_1**-min**	0.542 (0.114) λ=0.001	0.71 (0.043) λ=0.001	0.792 (0.032) λ=1	0.781 (0.081) λ=5	0.384 (0.068) λ=0.001	0.81 (0.041) λ=10	0.878 (0.041) λ=1	0.866 (0.054) λ=0.001	0.788 (0.14) λ=0.001	0.833 (0.095) λ=0.001	0.994 (0.009) λ=0.001	0.947 (0.021) λ=0.001	0.715 (0.054) λ=10	0.823 (0.01) λ=100
ℓ_2**-min**	0.542 (0.114) λ = 0.001	0.71 (0.043) λ = 0.001	0.792 (0.032) λ = 1	0.788 (0.07) λ = 5	0.384 (0.068) λ = 0.001	0.815 (0.04) λ = 10	0.876 (0.042) λ = 3	0.866 (0.054) λ = 0.001	0.788 (0.14) λ = 0.001	0.834 (0.094) λ = 0.001	0.994 (0.009) λ = 0.001	0.947 (0.021) λ = 0.001	0.728 (0.054) λ = 10	0.827 (0.009) λ = 100
ℓ_1**-max**	0.777 (0.061) λ = 3	0.806 (0.034) λ = 3	0.817 (0.029) λ = 0.5	0.733 (0.013) λ = 0.001	0.443 (0.067) λ = 10	0.724 (0.029) λ = 0.01	0.845 (0.039) λ = 0.001	0.959 (0.025) λ = 0.5	0.841 (0.09) λ = 100	0.89 (0.06) λ = 10	0.994 (0.009) λ = 0.001	0.96 (0.016) λ = 0.1	0.701 (0.004) λ = 5	0.798 (0.007) λ = 0.001
ℓ_2**-max**	0.776 (0.061) λ =100	0.803 (0.033) λ = 100	0.813 (0.03) λ = 100	0.733 (0.013) λ = 100	0.445 (0.065) λ = 1000	0.724 (0.029) λ = 0.001	0.844 (0.038) λ = 0.001	0.959 (0.027) λ=5	0.841 (0.09) λ = 1000	0.887 (0.06) λ = 5	0.994 (0.009) λ = 0.001	0.958 (0.016) λ = 0.5	0.701 (0.004) λ = 1000	0.798 (0.007) λ = 0.001
ℓ_1**-mean**	0.542 (0.114) λ = 0.001	0.71 (0.043) λ = 0.001	0.809 (0.031) λ = 3	0.779 (0.054) λ = 3	0.39 (0.071) λ = 0.1	0.844 (0.029) λ = 5	0.866 (0.042) λ = 1	0.942 (0.03) λ = 3	0.788 (0.14) λ = 0.001	0.834 (0.096) λ = 0.001	0.994 (0.009) λ = 0.001	0.96 (0.017) λ = 0.5	0.776 (0.03) λ = 5	0.844 (0.009) λ = 5
ℓ_2**-mean**	0.542 (0.114) λ.= 0.001	0.71 (0.043) λ = 0.001	0.808 (0.031) λ = 10	0.78 (0.053) λ = 100	0.384 (0.068) λ = 0.001	0.843 (0.029) λ = 1000	0.866 (0.042) λ = 3	0.938 (0.034) λ = 100	0.788 (0.14) λ = 0.001	0.834 (0.094) λ = 0.001	0.994 (0.009) λ = 0.001	0.963 (0.017) λ.= 1.0	0.776 (0.028) λ = 1000	0.844 (0.009) λ = 1000
ℓ_1**-LOS**	0.542 (0.114) λ = 0.001	0.71 (0.043) λ = 0.001	0.791 (0.032) λ = 0.001	0.733 (0.013) λ = 0.001	0.384 (0.068) λ = 0.001	0.724 (0.029) λ = 0.001	0.845 (0.039) λ = 0.001	0.866 (0.054) λ = 0.001	0.788 (0.14) λ = 0.001	0.834 (0.096) λ = 0.001	0.994 (0.009) λ = 0.001	0.947 (0.021) λ = 0.001	0.696 (0.016) λ = 0.001	0.798 (0.007) λ = 0.001
ℓ_2**-LOS**	0.63 (0.103) λ = 1000	0.712 (0.043) λ = 1	0.807 (0.032) λ = 10	0.733 (0.013) λ = 10	0.427 (0.069) λ = 100	0.831 (0.029) λ = 1000	0.865 (0.043) λ = 1000	0.900 (0.045) λ = 5	0.824 (0.099) λ = 1000	0.856 (0.075) λ = 5	0.994 (0.009) λ = 0.001	0.961 (0.017) λ = 3	0.771 (0.027) λ = 1000	0.842 (0.009) λ = 100
Simple-min	NA	NA	0.781 (0.034)	0.735 (0.11)	NA	0.81 (0.042)	0.839 (0.054)	NA	NA	NA	0.988 (0.015)	0.901 (0.038)	0.7 (0.059)	0.823 (0.01)
Simple-max	0.774 (0.06)	0.796 (0.035)	0.812 (0.027)	0.722 (0)	0.445 (0.065)	0.702 (0)	0.727 (0.016)	0.945 (0.03)	0.841 (0.09)	0.89 (0.06)	0.8 (0)	0.785 (0.003)	0.701 (0.004)	0.791 (0)
Simple-mean	0.417 (0.107)	0.7 (0.05)	0.809 (0.031)	0.753 (0.025)	NA	0.844 (0.029)	0.859 (0.04)	0.938 (0.035)	NA	0.705 (0.129)	0.965 (0.023)	0.949 (0.018)	0.776 (0.029)	0.844 (0.009)
AdaBoostM1	0.692 (0.082)	0.702 (0.052)	0.808 (0.028)	0.723 (0.096)	NA	0.823 (0.039)	0.767 (0.068)	0.893 (0.049)	0.668 (0.166)	0.85 (0.091)	0.969 (0.023)	0.906 (0.024)	0.742 (0.034)	0.772 (0.021)
LogitBoost	0.712 (0.076)	0.657 (0.047)	0.807 (0.032)	0.728 (0.108)	NA	0.839 (0.034)	0.764 (0.053)	0.742 (0.095)	NA	0.84 (0.083)	0.976 (0.023)	0.925 (0.026)	0.783 (0.032)	0.806 (0.019)
GentleBoost	0.713 (0.078)	0.707 (0.041)	0.807 (0.03)	0.744 (0.1)	NA	0.827 (0.041)	0.763 (0.059)	0.804 (0.098)	NA	0.871 (0.081)	0.978 (0.02)	0.925 (0.02)	0.794 (0.031)	0.784 (0.027)
RUS-Boost	0.729 (0.045)	0.651 (0.037)	0.788 (0.031)	0.674 (0.096)	0.49 (0.032)	0.756 (0.059)	0.759 (0.06)	0.634 (0.14)	0.489 (0.107)	0.822 (0.085)	0.952 (0.034)	0.881 (0.022)	0.721 (0.035)	0.737 (0.035)
Subspace	0.318 (0.109)	0.524 (0.021)	0.633 (0)	0.57 (0.095)	0.38 (0.012)	0.702 (0.004)	0.554 (0.078)	0.504 (0.024)	0.355 (0.111)	0.67 (0.132)	0.863 (0.042)	0.789 (0.033)	0.702 (0.003)	0.795 (0.004)
Bag	0.689 (0.073)	0.734 (0.047)	0.769 (0.034)	0.716 (0.099)	0.362 (0.071)	0.728 (0.071)	0.746 (0.072)	0.899 (0.045)	NA	0.869 (0.068)	0.958 (0.028)	0.921 (0.029)	0.776 (0.028)	0.801 (0.012)
MKLGL	0.761 (0.065)	0.79 (0.037)	0.783 (0.031)	0.78 (0.078)	0.329 (0.07)	0.839 (0.031)	0.846 (0.047)	0.955 (0.026)	0.784 (0.134)	0.887 (0.059)	0.994 (0.009)	0.965 (0.016)	0.76 (0.031)	0.841 (0.009)

Each cell presents the F1-score—mean (std-dev), and the corresponding best regularization parameter λ (when relevant), based on 100 experimental trials.
*Cells marked with NA indicate that the algorithm learned to classify all instances as a negative class, i.e., for any input the classifier returns -1. Based on the definition of the F_1 score in (5.24), precision is undefined if a classifier learns to classify all instances as the negative class.

Chapter 6

Online learning of the Fuzzy Choquet Integral for Decision-level fusion

6.1 Introduction

Chapter 5 introduced a data support-based training approach for learning the FM for decision-level fusion. A QP-based approach was used to solve the error minimization problem to learn the FM. However, since the number of parameters in an FM scales as 2^m, where m is the number of classifiers that are being combined, the QP-based approach (i.e., batch method) involves training 2^m parameters subject to a constraint matrix of size $(m(2^{m-1}-1)) \times (2^m - 1)$. For large values of m, i.e., $m > 6$, the space complexity is so

high that it becomes impractical to learn the ChI using the batch method. This has limited the application of our ChI-based models to fusion of a limited number ($m \leq 6$) of sources.

This chapter introduces an online learning approach for training the FM for aggregation using the ChI. This is an iterative gradient descent approach where the training data are processed one observation at a time. This greatly reduces the space complexity of the algorithm compared to the batch approach and thus extends the applicability of ChI-based aggregation to fusion of larger numbers of input sources.

To evaluate the performance of our online approach, we performed a set of experiments using synthetic and real-world data sets. The online algorithm's performance was compared against our previously proposed QP-based method. Since this online approach is an iterative process, we also looked at the convergence behavior compared to the batch method.

6.2 Online Learning Algorithm

Consider a set of training data $H = \{\mathbf{h}_1, \mathbf{h}_2, \ldots\}$, $\mathbf{h}_i \in \mathcal{R}^N$, where $\mathbf{h}_i$ is the ith training data instance. At each step, we are presented with a new training example $\mathbf{h}_t$ $t = 1, 2, \ldots$. For a given training example $\mathbf{h}$, it can be show that ChI in (1.2) can be reformulated as

[111]

$$C_g(\mathbf{h}) = \sum_{j=1}^{N} g(A_j)(h_{\pi(j)} - h_{\pi(j+1)}), \tag{6.1}$$

where $h_{\pi(N+1)} = 0$. let $\mathbf{u}$ be a vectorized form of the FM, e.g., in lexicographic or binary order. With slight manipulation, we can express (6.1) as a linear vector equation,

$$C_{\mathbf{u}}(\mathbf{h}) = {}_{\pi}\mathbf{h}^T \mathbf{u}_{\pi}, \tag{6.2}$$

where ${}_{\pi}\mathbf{h} = (h_{\pi(1)}, (h_{\pi(2)} - h_{\pi(1)}), \ldots, (h_{\pi(N)} - h_{\pi(N-1)}))^T$, and $\mathbf{u}_{\pi} = (g(x_{\pi(1)}), g(\{x_{\pi(1)}, x_{\pi(2)}\}), \ldots, g(X))^T$.

Let $\mathbf{y} = \{y_1, \ldots, y_M\}$ be the training data labels corresponding to each of $\mathbf{h}_i$. We can now express the learning of the FM $\mathbf{u}$ as an optimization of squared error with respect to the training labels (target variable) and (6.2).

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} E^2((\mathbf{y}, H), \mathbf{u}) = \arg\min_{\mathbf{u}} \sum_{i=1}^{M} \left(y_i - C_{\mathbf{u}}(\mathbf{h}_i)\right)^2, \tag{6.3}$$

subject to the boundary and monotonicity constraints on $\mathbf{u}$. We wish to solve this in an online fashion; hence, we apply a gradient descent step for each presented training example $\mathbf{h}_t$,

$$\mathbf{u}^{t+1} = \mathbf{u}^t - \eta^t \vec{\Delta}, \quad t = 1, \ldots, M, \tag{6.4}$$

where η^t is the learning rate and $\vec{\Delta}$ is the update vector. To do this, we will use the proximal

gradient method. First, we express the squared error with respect to (6.2) as

$$\begin{aligned} E^2((y_t, \mathbf{h}_t), \mathbf{u}^t) &= (y_t - C_{\mathbf{u}^t}(\mathbf{h}_t))^2 \\ &= y_t^2 - 2y_t \cdot {}_\pi\mathbf{h}_t^T \mathbf{u}_\pi^t + (\mathbf{u}_\pi^t)^T {}_\pi\mathbf{h}_t^T {}_\pi\mathbf{h}_t \mathbf{u}_\pi^t, \end{aligned} \tag{6.5}$$

which reinforces our assertion that the result of the ChI for a given training instance $\mathbf{h}_t$ only depends on the N values of $\mathbf{u}_\pi$. Hence, we can take the gradient of $E^2((y_t, \mathbf{h}_t), \mathbf{u}^t)$ with respect to $\mathbf{u}_\pi^t$,

$$\nabla_{\mathbf{u}_\pi^t} E^2((y_t, \mathbf{h}_t), \mathbf{u}^t) = -2y_{t\pi}\mathbf{h}_t + {}_\pi\mathbf{h}_t^T {}_\pi\mathbf{h}_t \mathbf{u}_\pi^T = \Delta_t, \tag{6.6}$$

where this becomes the update step in the sequential learning equation,

$$\tilde{\mathbf{u}}_\pi^{t+1} = \mathbf{u}_\pi^t - \eta_t \Delta_t. \tag{6.7}$$

However, this update does not consider the monotonicity and boundary constraints on the FM $\mathbf{u}$. To address this, the elements $\tilde{\mathbf{u}}_\pi^{t+1}$ are placed back into $\mathbf{u}$ appropriately; we call this intermediate variable $\tilde{\mathbf{u}}^{t+1}$. We now (may) need to project $\tilde{\mathbf{u}}^{t+1}$ back into the space defined by the constraints. We present two methods for this projection: i) a min-distance projection and ii) a practical adjustment method.

6.2.1 Min-distance projection

The vector $\tilde{\mathbf{u}}^{t+1}$ perhaps already lives in the space defined by the constraints on $\mathbf{u}$ and, if it does, then $\mathbf{u}^{t+1} = \tilde{\mathbf{u}}^{t+1}$. But if $\tilde{\mathbf{u}}^{t+1}$ violates any constraint, we project it back into the constraint space. The first method to address this is to project $\tilde{\mathbf{u}}^{t+1}$ by

$$\mathbf{u}^{t+1} = \text{Proj}_{A_{\mathbf{u}} \geq 0,\ \mathbf{u}_N = 1} \|\mathbf{u} - \tilde{\mathbf{u}}^{t+1}\|^2, \tag{6.8}$$

which simply projects $\tilde{\mathbf{u}}^{t+1}$ back in to the constraint space while minimizing the distance between the projection $\mathbf{u}$ and $\tilde{\mathbf{u}}^{t+1}$. This is accomplished by a simple linear constraint QP,

$$\mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} \mathbf{u}^T\mathbf{u} - 2(\tilde{\mathbf{u}}^{t+1})^T\mathbf{u},\ A\mathbf{u} \geq 0, \mathbf{u}_N = 1. \tag{6.9}$$

The challenge with (6.9) is that A is large, albeit very sparse and composed only of the values $\{0, 1, -1\}$.

6.2.2 Practical adjustment method

The second method to project $\tilde{\mathbf{u}}^{t+1}$ to the constraint space is to adjust values of $\tilde{\mathbf{u}}^{t+1}$ that violate the constraints incrementally. We accomplish this by applying the following adjustments, on a layer-by-layer basis,

$$u(A) = 1, \quad |A| = N, \tag{6.10a}$$

$$u(A) = \max\{u(A), u(A/x_j)\}, \quad \forall A, j : A \subset X, x_j \subset A, \tag{6.10b}$$

$$u(A) = \min\{u(A), u(A \cup x_j)\}, \quad \forall A, j : A \subset X, x_j \not\subset A, \tag{6.10c}$$

These updates are performed for each layer of the FM, which ensures that the overall monotonicity is maintained. In practice, a random permutation of the N layers is drawn and then (6.10) are applied in the permuted sequence per layer.

6.3 Synthetic Data Experiments

We designed a synthetic data experiment to assess the ability of our online algorithm to accurately learn the underlying FM. For each experiment, we randomly generated a *ground-truth*-FM (GT-FM) using the algorithm presented in [62]. A synthetic 6-dimensional data

set (six data columns of random real numbers generated from a uniform distribution between 0 and 1 was aggregated through this GT-FM to produce an output representing the target value. This 6-dimensional data set and the corresponding ground-truth target values were used for training and testing a new FM. In this experiment, the practical adjustment method presented in Section 6.2.2 was used for projecting the intermediate FM vector $\mathbf{u}$ to the constraint space. We trained the new FM using 80% of the total (1,000) observations, and evaluated its performance using the remaining 20%. The MSE between the ground-truth target values and the values produced by the learned FM is used as the performance measure. Fig. 6.1 shows the performance of the FM trained using the online method. The MSE consistently went under 10^{-4} within 20 epochs of training. This experiment tests the ability of our learning algorithm to accurately learn the FM that originally produced the training data; hence, we expect very good performance. This essentially is a sanity check, and the results indicate a quick convergence of the trained FM with the underlying ground-truth FM.

6.4 Experiments on real data

We also evaluated our online learning method on real world data sets from the UCI Machine Learning repository [33]. We chose 12 classification data sets that cover a wide range of decision problems. These data sets include the areas of health care, biology, material science, environmental science, and physics. Also, the selected data sets vary widely in the

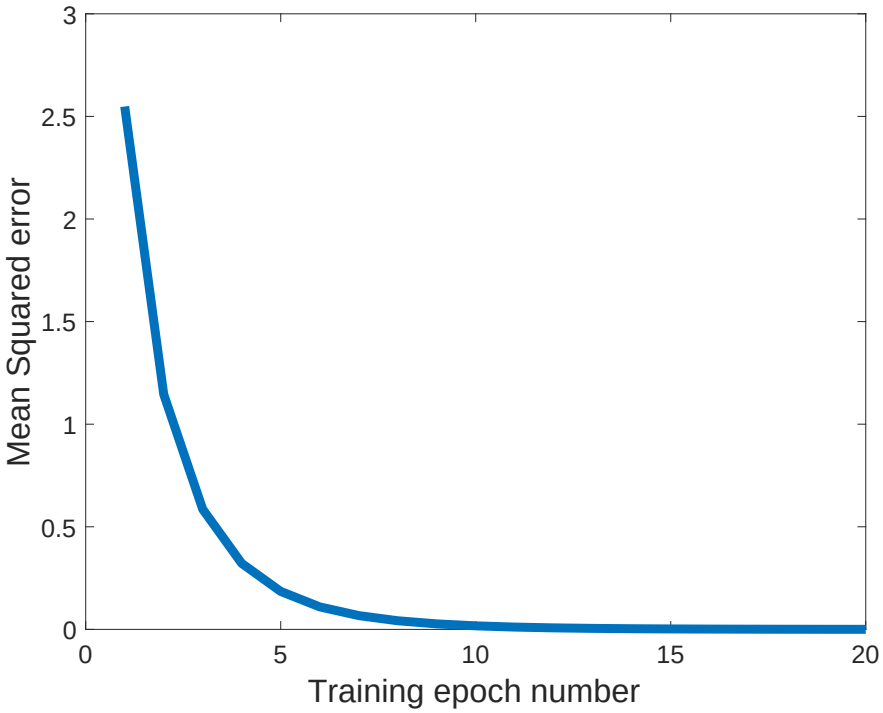

Figure 6.1: Performance of the online training algorithm on synthetic training data. MSE is measured between the ground-truth target values of the synthetic data and the values predicted by the FM learned using the online algorithm.

number of features and the number of observations, adding to the diversity. Each experiment consists of 10 random cross-fold trials, where in each trial a random partition of 80% of the data is used for training and the remaining data are used for testing. The classification performance is measured using F_1 score. F_1 score is the harmonic mean of $precision$ and $recall$; see Section 5.2.6.2 for details on the F_1 score. A perfect $F1$ score is 1, while 0 indicates poor performance. The results we report comprise the mean and standard deviation of F_1 score on the testing data. For the online method, since it is an iterative gradient descent approach, we initiate the FM vector with a set of values. We tried three different types of initialization—min initialization where all the FM values are 0s (except $u(X) = 1$, due to boundary constraints), the mean initialization when $u(A_i) = |A_i|/n$, $\forall A_i \subset X$, and the

max initialization where the FM is all 1s (except $u(\emptyset) = 0$, due to boundary constraints).

6.4.1 Results

Table 6.1 summarizes the results of our experimental analysis to compare the performance of our online method with the batch method. Here, we used 6 SVM kernels for both the batch and the online methods. We performed two-tailed t-tests to identify the best performing methods on each data set at a 5% significance level, and such better performers were marked in bold in Table 6.1. For 10 out of 12 data sets, at least one our online methods have either matched or outperformed the batch method. Unlike the batch method which is limited to fusion of 6 or less sources, the online method can be extended beyond 6 sources. We tested the online method on the same data sets with 8 and 10 SVM kernels (input sources) to evaluate any additional gains. While the online method with 8 SVM kernels shows marginal gains over 6 kernels, the 10-kernel version demonstrates some significant gains over the base 6-kernel version presented in Table 6.1. The comparison of batch method with 10-SVM kernel online method is summarized in Table 6.2. The base 6-SVM kernel online method outperformed the batch method on only two data sets (Heart and Sonar), The 10-SVM kernel online method outperformed the batch method on 9 out of 12 data sets. Note that the QP (batch method) is unable to fuse 10-SVM kernels due to space complexity limitations.

We visualized the performance of the FMs learned with the online method to examine the iterative performance trend compared with the static batch method. Fig. 6.2 shows the iterative F_1 score for the Ionosphere data set on the testing data. Similarly, Figs. 6.3 and 6.4 show the iterative performance for Mmass and Vertebral data sets, respectively. On most data sets, the F_1 score from the online method converges within 50 training epochs. Initialization of the FM does play a noticeable role in the convergence pattern of the online method. For max and min initialized instances, the F_1 score of the online method started relatively farther from the mean F_1 achieved by the static batch method, and converged to the batch method results over several epochs of training. Contrary to this, when using the mean FM initialization, as shown in Fig. 6.3 on the Mmass data set, the online F_1 score converged quite close to the batch method with just one epoch of training. For some data sets like Vertebral in Fig. 6.4, the online F_1 score starts higher and eventually converged to the batch method F_1 score. This pattern indicates that QP-based batch method is converging to a sub-optimal solution, likely over-fit to the training data. For such cases, the online method enables us to halt the training process at an appropriate point to extract the FM with best testing performance (assuming a validation data set is available).

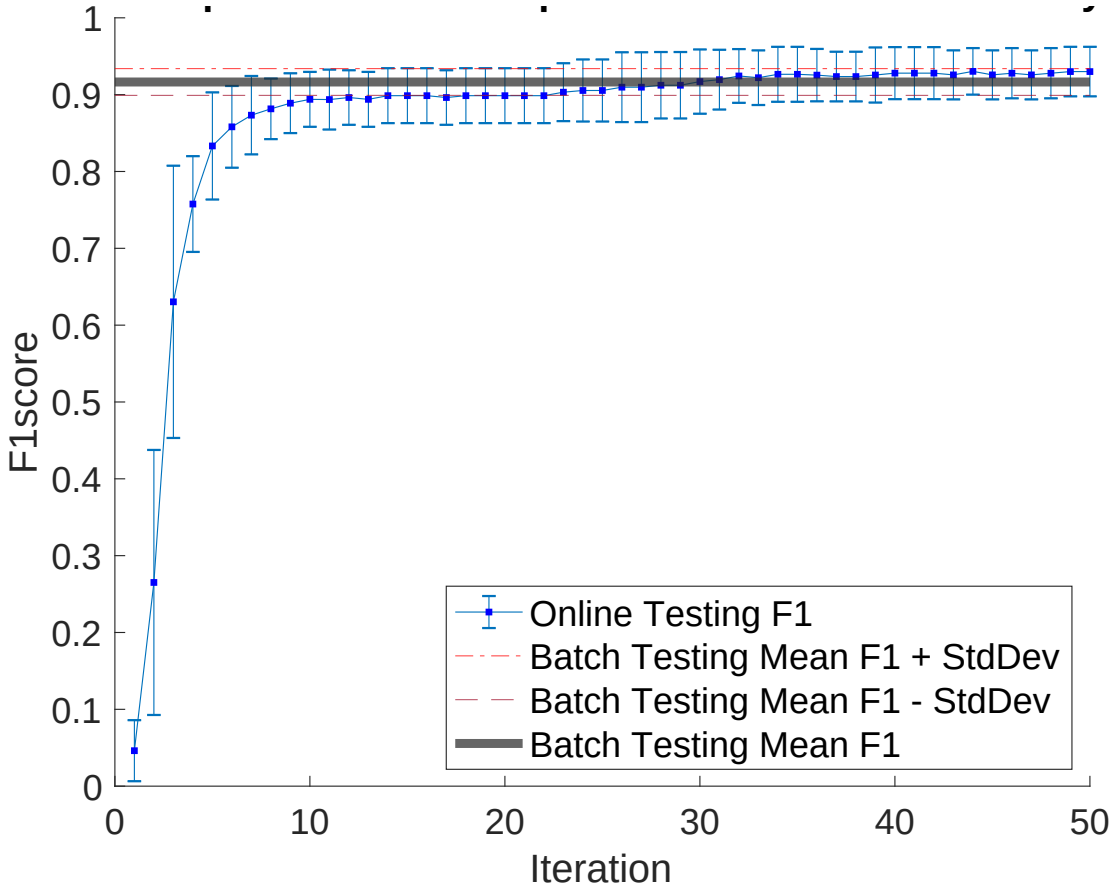

Figure 6.2: Iterative comparison of the online method with the batch method on Ionosphere data set. The online method was initiated with a max aggregation FM. Six SVM kernels were used for both the online and the batch methods.

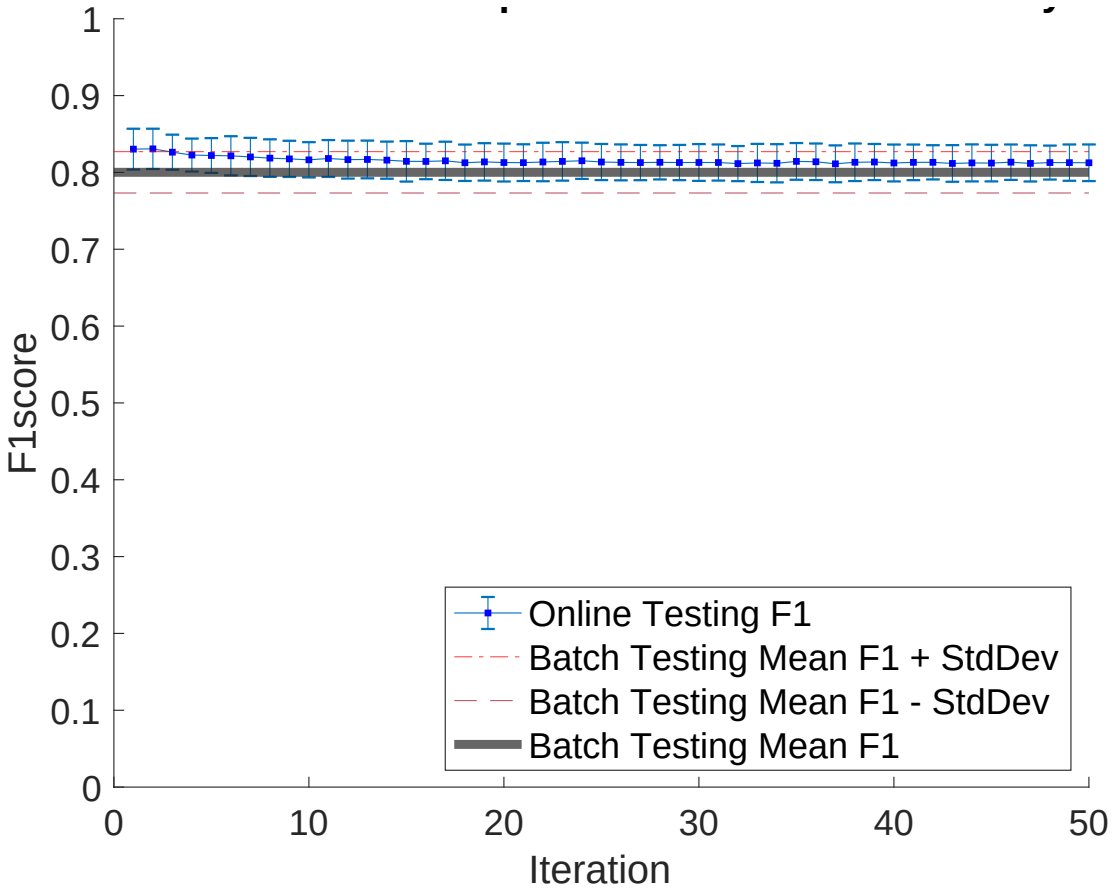

Figure 6.3: Iterative comparison of the online method with the batch method on Mmass data set. The online method was initiated with a mean aggregation FM. 10 SVM kernels were used for the online method while the batch method has used 6 SVM kernels.

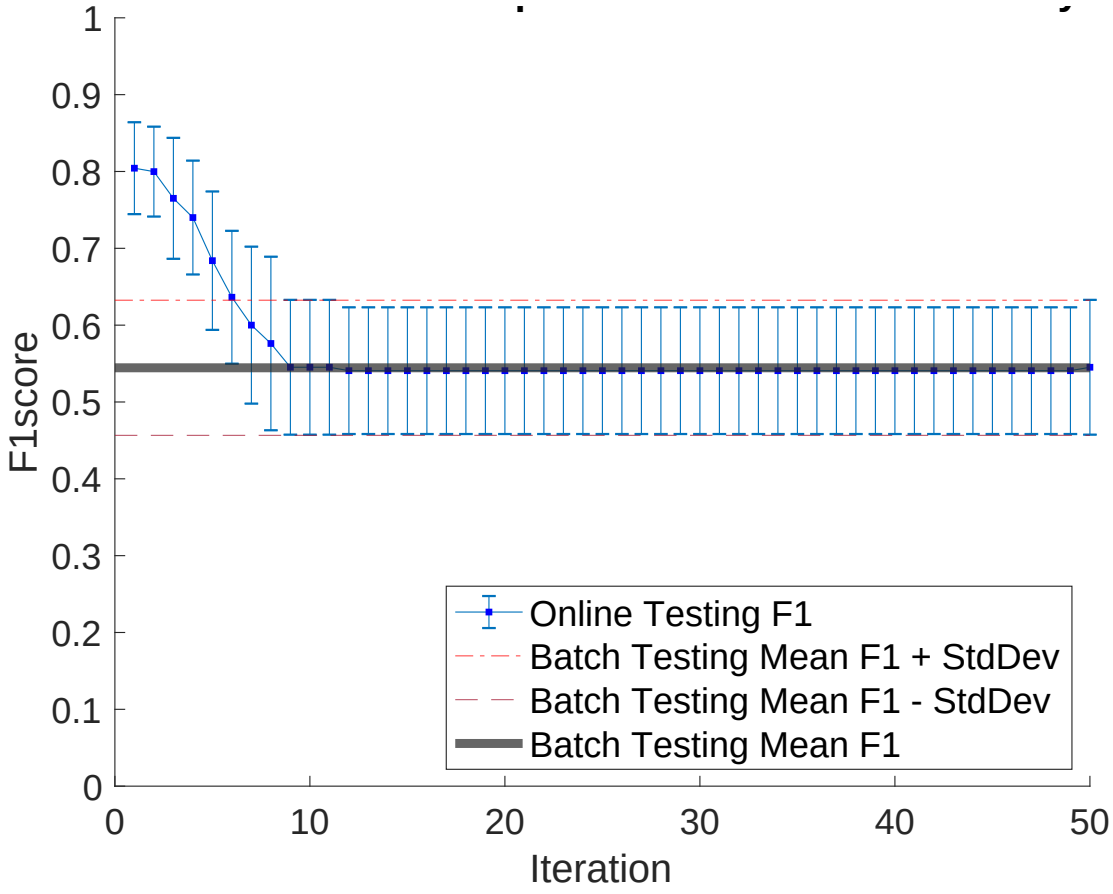

Figure 6.4: Iterative comparison of the online method with the batch method on Vertebral data set. The online method was initiated with a max aggregation FM. Six SVM kernels were used for both the online and the batch methods.

Table 6.1
Performance comparison of online and batch methods on real-world data (6-SVM kernels).

	Data Set											
Method-(initialization*)	Vertebral	Biodeg	Mmass	Breast-Cancer	Trans-fusion	Heart	Sonar	Derma-tology	Ecoli	Glass	Iono-sphere	Absentee
QP (batch method)	**0.531 (0.099)**	**0.715 (0.041)**	**0.788 (0.031)**	**0.735 (0.015)**	**0.370 (0.081)**	0.732 (0.037)	0.841 (0.032)	**0.861 (0.057)**	**0.774 (0.143)**	**0.868 (0.068)**	**0.942 (0.031)**	**0.697 (0.016)**
Online (Min)	0.000 (0.000)	0.205 (0.076)	**0.786 (0.028)**	**0.735 (0.015)**	0.005 (0.017)	**0.811 (0.031)**	**0.891 (0.031)**	**0.858 (0.056)**	0.000 (0.000)	0.610 (0.131)	0.918 (0.036)	**0.705 (0.019)**
Online (Mean)	0.418 (0.094)	0.650 (0.053)	**0.789 (0.031)**	0.722 (0.000)	**0.309 (0.092)**	0.785 (0.029)	**0.854 (0.043)**	0.861 (0.057)	0.506 (0.189)	**0.819 (0.074)**	**0.944 (0.031)**	**0.705 (0.022)**
Online (Max)	**0.530 (0.097)**	0.662 (0.032)	**0.789 (0.031)**	0.722 (0.000)	**0.380 (0.084)**	0.758 (0.038)	0.774 (0.014)	**0.867 (0.044)**	0.615 (0.193)	**0.872 (0.082)**	**0.949 (0.032)**	**0.699 (0.017)**

Each cell presents the F_1-score—mean (std-dev) based on 10 experimental trials. Cells marked in bold indicate best performing algorithms on each data set based on a two-tailed t-test at a 5% significance level. *The initialization used for the online learning algorithm (min/mean/max).

Table 6.2
Performance of 10-SVM kernel online method vs. 6-SVM kernel batch method on real-world data.

	Data Set											
Method-(initialization*)	Vertebral	Biodeg	Mmass	Breast-Cancer	Trans-fusion	Heart	Sonar	Derma-tology	Ecoli	Glass	Iono-sphere	Absentee
QP (batch method)	0.531 (0.099)	**0.715 (0.041)**	0.788 (0.031)	0.735 (0.015)	0.370 (0.081)	0.732 (0.037)	0.841 (0.032)	0.861 (0.057)	0.774 (0.143)	**0.868 (0.068)**	**0.942 (0.031)**	0.697 (0.016)
Online (Min)	0.000 (0.000)	0.019 (0.076)	0.787 (0.028)	**0.774 (0.015)**	0.000 (0.017)	**0.786 (0.031)**	**0.871 (0.031)**	0.100 (0.056)	0.000 (0.000)	0.000 (0.131)	0.905 (0.036)	0.683 (0.019)
Online (Mean)	0.296 (0.094)	0.568 (0.053)	**0.805 (0.031)**	0.749 (0.000)	0.222 (0.092)	**0.812 (0.029)**	**0.892 (0.043)**	0.898 (0.057)	0.431 (0.189)	0.742 (0.074)	**0.950 (0.031)**	**0.774 (0.022)**
Online (Max)	**0.677 (0.097)**	0.610 (0.032)	**0.812 (0.031)**	0.722 (0.000)	**0.442 (0.084)**	0.703 (0.038)	0.730 (0.014)	**0.938 (0.044)**	**0.845 (0.193)**	**0.863 (0.082)**	0.802 (0.032)	0.696 (0.017)

Each cell presents the F_1-score—mean (std-dev) based on 10 experimental trials. Cells marked in bold indicate best performing algorithms on each data set based on a two-tailed t-test at a 5% significance level. *The initialization used for the online learning algorithm (min/mean/max).

6.5 Conclusions and Future Work

This chapter proposed an online learning approach for training the ChI-based decision fusion models. We compared the performance of the online method with our previously proposed QP-based batch training approach. In our experimental evaluation, the online method has matched or outperformed the batch method on 10 out of 12 data sets. The batch method, due to its exorbitant space complexity, limits the application of ChI-based decision fusion to 6 or fewer input sources. The introduction of the online method allows us to extend the application of ChI for decision fusion to more than 6 sources. Our experiments fusing 8 and 10 sources showed superior performance over the batch method (with 6 sources).

Future work will focus on enhancing the online method by adding regularization to the learning approach. In our recent prior work [69], we introduced a goal-based regularization strategy to train the FMs for decision fusion using ChI. We plan to extend the goal-based regularization to our online learning approach.

Chapter 7

Conclusion

This dissertation presented novel feature- and decision-level aggregation methods. In addition to improved performance, the proposed methods have also addressed the problem of over-fitting through the application of novel regularization approaches. The proposed visualization strategies helped to tease apart the interactions among the input variables, and to holistically examine the learned models as well as the model outputs. The online methods introduced in this dissertation have enabled the applicability of ChI data fusion methods to higher dimensional data sets. The key challenges addressed in this dissertation are summarized next.

Over-fitting

The ChI-based feature- and decision-level fusion methods introduced in this work have outperformed the traditional aggregation methods like linear regression and linear order statistic. However, since the FM of ChI scales as 2^m where m is the number of input sources, learning ChI-based algorithms require a relatively large amount of training data (though perhaps not nearly as much as a deep learning method). Limited amount of training data often results in over-fit solutions. This work has addressed the problem of over-fitting by introducing new regularization functions that enable the user to encode knowledge of the underlying FM structure into the algorithm's training procedure. Using these regularizers, the user can define a particular goal for the FM before learning. The experimental results using synthetic and real-world data sets indicated that the addition of goal-based regularizers proposed in this work results in statistically significant improvements in the performance of the algorithms.

Explainability

In many real-world applications, it is imperative to have an explanation. This explanation may include information on what was learned, what is the worth of individual input sources, why a decision was reached, and what evidence process(es) were used to reach that decision. Most machine learning solutions for data fusion are "black boxes," e.g., deep learning. The visualization techniques and evaluation measures presented in this work allow the user to explain the learned solution as well as the outputs produced by the models.

This work has also presented the visualization strategies and metrics to assess the data diversity and the impact of regularization on the learned model.

Space complexity of the training algorithm

The ChI data fusion methods presented in chapters 3 and 5 demonstrated superior performance over traditional aggregation approaches like linear regression and linear order statistic. However, since the number of learned parameters for ChI-based methods grows exponentially with the number of input sources, the exorbitant space complexity for learning the models on higher dimensional data sets has limited the practical applicability of these methods to fusion of six or fewer input sources. To address this, this work introduced an online learning method for ChI-based data fusion models. This is an iterative gradient descent approach that processes one observation at a time, and thus requires less computation and memory during the training. The online method enabled the applicability of ChI-based data fusion methods on higher dimensional data sets.

7.1 Future work

The following are the potential avenues for the future work on the problems addressed in this dissertation.

Goal-based regularization of the online learning algorithms

The online learning algorithms introduced in this dissertation expanded the applicability of ChI-based data fusion methods to data sets with higher dimensionality than possible earlier. However, since the number of trainable parameters of the ChI grows exponentially with the number of input sources, the problem of over-fit models could be more acute when dealing with higher dimensional data sets. Developing regularization approaches for the online training of ChI is an important open problem. Extending ℓ_1, ℓ_2, and goal-based regularization methods presented in chapters 3 and 5 to the online version of the learning algorithm could greatly enhance the applicability of the ChI-based data fusion to a wider range of applications and problems.

ChI-based deep learning models and explainability

The machine learning models based on deep learning architectures have made some significant leaps in numerous applications. However, most deep learning models involve a complex network architecture with a large set of trainable parameters, turning them into

"black box" solutions. There are current collaborators who are exploring the application of explainable ChI-based models on deep learning architectures. This strategy can now take advantage of the regularization strategies and online learning methods introduced in this dissertation to extend the training of ChI-based deep learning models to higher dimensional data sets at a reduced space complexity.

Modeling the uncertainty of ChI-based aggregation models

The number of trainable parameters of the FM of ChI scales exponentially with the number of input sources. The volume and the variety of the training data plays a crucial role in the quality of the learned models. When a training data set does not contain sufficient number of observations for all the possible sort orders (i.e., data set is less diverse), an FM learned from such a data set will have a portion of "untouched" FM nodes that are set just based on the monotonicity and boundary constraints. This results in an inherent uncertainty of the learned model. The goal-based regularization strategies presented in this work control such uncertainty by pushing the "untouched" FM nodes towards a predefined goal. However, when no such predefined goal is known, it is valuable to model the inherent uncertainty of the model and produce interval-valued outputs with corresponding confidence scores. Future work could explore the development of such ChI-based models for data fusion under uncertainty. Future work could also explore the application of explainable indices such as *Shapley* and *Interaction*, on interval-valued FMs to explain different aspects of the model uncertainty.

References

[1] O. Akbilgic, H. Bozdogan, and M. E. Balaban. A novel hybrid rbf neural networks model as a forecaster. *Statistics and Computing*, 24(3):365–375, 2014.

[2] G. R. Amin and S. K. Sharma. Measuring batting parameters in cricket: A two-stage regression-owa method. *Measurement*, 53:56–61, 2014.

[3] D. Anderson, T. Havens, C. Wagner, J. Keller, M. Anderson, and D. Wescott. Extension of the fuzzy integral for general fuzzy set-valued information. *IEEE Trans. on Fuzzy Systems*, 22(6):1625–1639, Dec 2014.

[4] D. Anderson, S. Price, and T. Havens. Regularization-based learning of the choquet integral. In *IEEE Int. Conf. on Fuzzy Systems*, pages 2519–2526, July 2014.

[5] D. T. Anderson, J. M. Keller, and T. C. Havens. Learning fuzzy-valued fuzzy measures for the fuzzy-valued Sugeno fuzzy integral. In *Proc. IPMU*, Dortmund, Germany, 2010.

[6] D. T. Anderson, S. Price, and T. C. Havens. Regularization-based learning of the choquet integral. In *Proc. IEEE Int. Conf. Fuzzy Systems*, 2014.

[7] M. F. Anderson, D. T. Anderson, and D. J. Wescott. Estimation of adult skeletal age-at-death using the sugeno fuzzy integral. *American J. of physical anthropology*, 142(1):30–41, 2010.

[8] S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo. Assessing non-additive utility for multicriteria decision aid. *European J. of Operational Research*, 158(3):734–744, 2004.

[9] S. Angilella, S. Greco, and B. Matarazzo. Non-additive robust ordinal regression: A multiple criteria decision model based on the choquet integral. *European J. of Operational Research*, 201(1):277–288, 2010.

[10] S. Auephanwiriyakul, J. M. Keller, and P. D. Gader. Generalized choquet fuzzy integral fusion. *Information Fusion*, 3(1):69–85, 2002.

[11] I. Barandiaran. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):1–22, 1998.

[12] G. Beliakov. Construction of aggregation functions from data using linear programming. *Fuzzy Sets and Systems*, 160:65–75, 2009.

[13] G. Beliakov. A new type of fuzzy integrals for decision making based on bivariate symmetric means. *Int. J. of Intell. Systems*, 33(8):1660–1671, 2018.

[14] G. Beliakov, S. James, and G. Li. Learning choquet-integral-based metrics for semisupervised clustering. *IEEE Trans. on Fuzzy Systems*, 19(3):562–574, 2011.

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[17] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[18] T. Brooks, D. Pope, and M. Marcolini. Airfoil self-noise and prediction. 1218, 08 1989.

[19] T. F. Brooks, D. S. Pope, and M. A. Marcolini. Airfoil self-noise and prediction. 1989.

[20] H. Bustince, M. Galar, B. Bedregal, A. Kolesarova, and R. Mesiar. A new approach to interval-valued choquet integrals and the problem of ordering in interval-valued fuzzy set applications. *IEEE Trans. on Fuzzy systems*, 21(6):1150–1162, 2013.

[21] M. Cassotti, D. Ballabio, V. Consonni, A. Mauri, I. Tetko, and R. Todeschini. Prediction of acute aquatic toxicity toward daphnia magna by using the ga-knn method. *Alternatives to laboratory animals : ATLA*, 42:31–41, 03 2014.

[22] M. Cassotti, D. Ballabio, V. Consonni, A. Mauri, I. V. Tetko, and R. Todeschini. Prediction of acute aquatic toxicity toward daphnia magna by using the ga-k nn method. *Alternatives to Laboratory Animals*, 42(1):31–41, 2014.

[23] M. Cassotti, D. Ballabio, R. Todeschini, and V. Consonni. A similarity-based qsar model for predicting acute toxicity towards the fathead minnow (pimephales promelas). *SAR and QSAR in Environmental Research*, 26(3):217–243, 2015.

[24] C. C. Chang and C. J. Lin. LIBSVM: a library for support vector machines. *ACM Trans. Intell. Sys. Tech.*, 2(3):1–27, 2011.

[25] J.-H. Chiang. Choquet fuzzy integral-based hierarchical networks for decision analysis. *IEEE Trans. on Fuzzy Systems*, 7(1):63–71, 1999.

[26] F. Chiclana, F. Herrera, and E. Herrera-Viedma. The ordered weighted geometric operator: properties and application in mcdm problems. In *in Proc. 8th Conf. Inform. Processing and Management of Uncertainty in Knowledgebased Systems (IPMU*. Citeseer, 2000.

[27] G. Choquet. Theory of capacities. In *Annales de l'institut Fourier*, volume 5, pages 131–295. Institut Fourier, 1954.

[28] C. Cortes, M. Mohri, and A. Rostamizadeh. ℓ_2 regularization for learning kernels. In *Proc. Conf. Uncertainty in Artificial Intelligence*, pages 187–196, 2009.

[29] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[30] C. Cortes and V. N. Vapnik. Support-vector networks. volume 20, pages 273–297, 1995.

[31] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[32] B. V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997.

[33] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[34] G. P. Dimuro, J. Fernández, B. Bedregal, R. Mesiar, J. A. Sanz, G. Lucca, and H. Bustince. The state-of-art of the generalizations of the choquet integral: From aggregation and pre-aggregation to ordered directionally monotone functions. *Information Fusion*, 57:27–43, 2020.

[35] G. P. Dimuro, G. Lucca, B. Bedregal, R. Mesiar, J. A. Sanz, C.-T. Lin, and H. Bustince. Generalized cf1f2-integrals: From choquet-like aggregation to ordered directionally monotone functions. *Fuzzy Sets and Systems*, 378:44–67, 2020.

[36] X. Du and A. Zare. Multiple instance choquet integral classifier fusion and regression for remote sensing applications. *CoRR*, abs/1803.04048, 2018.

[37] D. J. Dubois. *Fuzzy sets and systems: theory and applications*, volume 144. Academic press, 1980.

[38] N. Feng, X. Yu, R. Dou, and B. Pan. Managing risk for business processes: A fuzzy based multi-agent system. *Journal of Intelligent & Fuzzy Systems*, 29(6):2717–2726, 2015.

[39] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of computer and system sciences*, 55(1):119–139, 1997.

[40] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[41] P. D. Gader, J. M. Keller, and B. N. Nelson. Recognition technology for the detection of buried land mines. *IEEE trans. on fuzzy systems*, 9(1):31–43, 2001.

[42] S. Garcia and F. Herrera. An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of machine learning research*, 9(Dec):2677–2694, 2008.

[43] J. Gerritsma, R. Onnink, and A. Versluis. Geometry, resistance and stability of the delft systematic yacht hull series. *International shipbuilding progress*, 28(328):276–297, 1981.

[44] J. Gerritsma, R. Onnink, and A. Versluis. Geometry, resistance and stability of the delft systematic yacht hull series. *International shipbuilding progress*, 28(328):276–297, 1981.

[45] M. Grabisch. Fuzzy integral in multicriteria decision making. *Fuzzy sets and Systems*, 69(3):279–298, 1995.

[46] M. Grabisch. Fuzzy integral in multicriteria decision making. *Fuzzy Sets and Systems*, 69:279–298, 1995.

[47] M. Grabisch. Fuzzy integral for classification and feature extraction. *Fuzzy Measures and Integrals: Theory and Apps.*, 1:415–434, 2000.

[48] M. Grabisch. Fuzzy integral for classification and feature extraction. In *Fuzzy Measures and Integrals: Theory and Applications*, pages 415–434. Springer-Verlag New York, Inc., 2000.

[49] M. Grabisch. *Fuzzy Measures and Integrals: Theory and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[50] M. Grabisch. Modelling data by the choquet integral. In V. Torra, editor, *Information Fusion in Data Mining*, volume 123 of *Studies in Fuzziness and Soft Compting*. Springer, Berlin, Heidelberg, 2003.

[51] M. Grabisch and C. Labreuche. A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1):247–286, 2010.

[52] M. Grabisch, T. Murofushi, and M. Sugeno, editors. *Fuzzy Measures and Integrals: Theory and Applications*. Physica-Verlag, New York, 2000.

[53] M. Grabisch, H. T. Nguyen, and E. A. Walker. *Fundamentals of Uncertainty Calculi, With Applications to Fuzzy Inference*. Kluwer Academic, Dordrecht, 1995.

[54] M. Grabisch, H. T. Nguyen, and E. A. Walker. *Fundamentals of uncertainty calculi with applications to fuzzy inference*, volume 30. Springer Science & Business Media, 2013.

[55] M. Grabisch and J.-M. Nicolas. Classification by fuzzy integral: Performance and tests. *Fuzzy sets and systems*, 65(2-3):255–271, 1994.

[56] M. Grabisch and M. Roubens. Application of the choquet integral in multicriteria decision making. *Fuzzy Measures and Integrals-Theory and Applications*, pages 348–374, 2000.

[57] M. Grabisch and M. Sugeno. Multi-attribute classification using fuzzy integral. In *[1992 Proc.] IEEE Int. Conf. on Fuzzy Systems*, pages 47–54. IEEE, 1992.

[58] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[59] T. C. Havens and D. T. Anderson. Machine learning of choquet integral regression with respect to a bounded capacity (or non-monotonic fuzzy measure). In *IEEE Int. Conf. Fuzzy Systems*, 2019.

[60] T. C. Havens, D. T. Anderson, and C. Wagner. Data-informed fuzzy measures for fuzzy integration of intervals and fuzzy numbers. *IEEE Transactions on Fuzzy Systems*, 23(5):1861–1875, 2014.

[61] 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. In *IEEE Int. Conf. on Fuzzy Systems*, pages 1–8. IEEE, 2013.

[62] T. C. Havens and A. J. Pinar. Generating random fuzzy (capacity) measures for data fusion simulations. In *2017 IEEE Symp. Ser. on Comp. Intell. (SSCI)*, pages 1–8. IEEE, 2017.

[63] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 6:65–70, 1979.

[64] L. Hu, D. T. Anderson, and T. C. Havens. Multiple kernel aggregation using fuzzy integrals. In *IEEE Int. conf. on Fuzzy Systems*, pages 1–7. IEEE, 2013.

[65] M. A. Islam, D. T. Anderson, A. Pinar, T. C. Havens, G. Scott, and J. M. Keller. Enabling explainable fusion in deep learning with fuzzy integral neural networks. *IEEE Trans. Fuzzy Sys.*, 2019.

[66] M. A. Islam, D. T. Anderson, A. J. Pinar, and T. C. Havens. Data-driven compression and efficient learning of the choquet integral. *IEEE Trans. on Fuzzy Systems*, 26(4):1908–1922, 2017.

[67] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

[68] S. Kakula, A. J. Pinar, T. C. Havens, M. A. Islam, and D. T. Anderson. Novel regularization for learning the fuzzy choquet integral with limited training data. *IEEE Trans. Fuzzy Sys.*, pages 1–6, 2020.

[69] S. K. Kakula, A. J. Pinar, T. C. Havens, and D. T. Anderson. Choquet integral ridge regression. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2020.

[70] J. Keller, P. Gader, and A. Hocaoglu. Fuzzy integral in image processing and recognition. In *Fuzzy Measures and Integrals: Theory and Applications*, pages 435–466. Springer-Verlag New York, Inc., 2000.

[71] J. Keller and J. Osborn. A reward/punishment scheme to learn fuzzy densities for the fuzzy integral. In *Int. Fuzzy Systems Assoc. World Congress*, pages 97–100, 1995.

[72] J. Keller and J. Osborn. Training the fuzzy integral. *Int. J. of Approximate Reasoning*, 15(1):1–24, 1996.

[73] J. M. Keller, P. Gader, H. Tahani, J.-H. Chiang, and M. Mohamed. Advances in fuzzy integration for pattern recognition. *Fuzzy sets and systems*, 65(2-3):273–283, 1994.

[74] M. Kloft, U. Brefeld, P. Laskov, and S. Sonnenburg. Non-sparse multiple kernel learning. In *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[75] L.-W. Ko, Y.-C. Lu, H. Bustince, Y.-C. Chang, Y. Chang, J. Ferandez, Y.-K. Wang, J. A. Sanz, G. P. Dimuro, and C.-T. Lin. Multimodal fuzzy fusion for enhancing the motor-imagery-based brain computer interface. *IEEE Comp. Intell. Magazine*, 14(1):96–106, 2019.

[76] C. Labreuche. Construction of a choquet integral and the value functions without any commensurateness assumption in multi-criteria decision making. In *Proc. of the 7th Conf. of the European Soc. for Fuzzy Logic and Tech.*, pages 90–97. Atlantis Press, 2011.

[77] H.-C. Liu, W.-S. Chen, Y.-C. Tu, and Y.-K. Yu. Choquet integral regression model based on high-order l-measure. In *Proc. Int. Conf. Mach. Learn. and Cyber.*, pages 3177–3182, July 2009.

[78] B. Llamazares. Constructing choquet integral-based operators that generalize weighted means and owa operators. *Information Fusion*, 23:131–138, 2015.

[79] R. Lourenzutti, R. A. Krohling, and M. Z. Reformat. Choquet based topsis and todim for dynamic and heterogeneous decision making with criteria interaction. *Information Sciences*, 408:41–69, 2017.

[80] G. Lucca, G. P. Dimuro, J. Fernández, H. Bustince, B. Bedregal, and J. A. Sanz. Improving the performance of fuzzy rule-based classification systems based on a nonaveraging generalization of cc-integrals named $c\_\{F\_1F\_2\}$-integrals. *IEEE Trans. on Fuzzy Systems*, 27(1):124–134, 2018.

[81] G. Lucca, J. A. Sanz, G. P. Dimuro, B. Bedregal, M. J. Asiain, M. Elkano, and H. Bustince. Cc-integrals: Choquet-like copula-based aggregation functions and its application in fuzzy rule-based classification systems. *Knowledge-Based Systems*, 119:32–43, 2017.

[82] G. Lucca, J. A. Sanz, G. P. Dimuro, B. Bedregal, H. Bustince, and R. Mesiar. Cf-integrals: A new family of pre-aggregation functions with application to fuzzy rule-based classification systems. *Information Sciences*, 435:94–110, 2018.

[83] G. Lucca, J. A. Sanz, G. P. Dimuro, B. Bedregal, R. Mesiar, A. Kolesarova, and H. Bustince. Preaggregation functions: Construction and an application. *IEEE Trans. on Fuzzy Systems*, 24(2):260–272, 2015.

[84] Matlab statistics and machine learning toolbox. *Release 2019b, The MathWorks, Inc., Natick, Massachusetts, United States.*

[85] A. Mendez-Vazquez and P. Gader. Sparsity promotion models for the choquet integral. In *IEEE Symp. on Foundations of Comp. Intell.*, pages 454–459. IEEE, 2007.

[86] A. Mendez-Vazquez, P. Gader, J. M. Keller, and K. Chamberlin. Minimum classification error training for choquet integrals with applications to landmine detection. *IEEE Trans. on Fuzzy Systems*, 16(1):225–238, 2008.

[87] F. Meng, H. Cheng, and Q. Zhang. Induced atanassov's interval-valued intuitionistic fuzzy hybrid choquet integral operators and their application in decision making. *Int. J. of Comp. Intell. Systems*, 7(3):524–542, 2014.

[88] J. M. Merigo and M. Casanovas. Decision-making with distance measures and induced aggregation operators. *Computers & Industrial Engineering*, 60(1):66–76, 2011.

[89] J. M. Merigó and M. Casanovas. Induced and uncertain heavy owa operators. *Computers & Industrial Engineering*, 60(1):106–116, 2011.

[90] T. Murofushi and S. Soneda. Techniques for reading fuzzy measures (iii): interaction index. In *9th fuzzy system symposium*, pages 693–696. Sapporo,, Japan, 1993.

[91] 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*, 29(2):201–227, 1989.

[92] T. Murofushi, M. Sugeno, and M. Machida. Non-monotonic fuzzy measures and the Choquet integral. *Fuzzy Sets and Systems*, 64:73–86, 1994.

[93] B. Murray, M. A. Islam, A. J. Pinar, D. T. Anderson, T. C. Havens, and G. Scott. Explainable ai for understanding decisions and data-driven optimization of the choquet integral. In *Proc. IEEE Int. Conf. Fuzzy Sys.*, 2018.

[94] Y. Narukawa, T. Murofushi, and M. Sugeno. Space of fuzzy measures and convergence. *Fuzzy sets and systems*, 138(3):497–506, 2003.

[95] Y. Narukawa and V. Torra. Non-monotonic fuzzy measures and intuitionistic fuzzy sets. In V. Torra, Y. Narukawa, A. Valls, and J. Domingo-Ferrer, editors, *Modeling Decisions for Artificial Intelligence*, volume 3885 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2006.

[96] P. Nemenyi. Distribution-free multiple comparisons (doctoral dissertation, princeton university). *Ann Arbor: University Microfilms*, (64-6278), 1963.

[97] A. Pinar, T. C. Havens, D. T. Anderson, and L. Hu. Feature and decision level fusion using multiple kernel learning and fuzzy integrals. In *IEEE Int. conf. on Fuzzy Systems*, pages 1–7, Aug 2015.

[98] 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*, 2(4):303–319, 2017.

[99] A. J. Pinar, T. C. Havens, M. A. Islam, and D. T. Anderson. Visualization and learning of the choquet integral with limited training data. In *IEEE Int. Conf. on Fuzzy Systems*, pages 1–6, July 2017.

[100] A. J. Pinar, T. C. Havens, M. A. Islam, and D. T. Anderson. Visualization and learning of the choquet integral with limited training data. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2017.

[101] 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 Transactions on Fuzzy Systems*, 25(6):1403–1416, 2016.

[102] S. Price, D. T. Anderson, C. Wagner, T. C. Havens, and J. M. Keller. Indices for introspection of the choquet integral. In *Studies in Fuzziness and Soft Computing*, volume 312 of *Proc. World Conf. Soft Computing*, pages 261–271, 2014.

[103] R. Sadiq, S. A. Haji, G. Cool, and M. J. Rodriguez. Using penalty functions to evaluate aggregation models for environmental indices. *Journal of environmental management*, 91(3):706–716, 2010.

[104] R. Sadiq and S. Tesfamariam. Probability density functions based weights for ordered weighted averaging (owa) operators: An example of water quality indices. *European Journal of Operational Research*, 182(3):1350–1368, 2007.

[105] R. Sadiq and S. Tesfamariam. Developing environmental indices using fuzzy numbers ordered weighted averaging (fn-owa) operators. *Stochastic Environmental Research and Risk Assessment*, 22(4):495–505, 2008.

[106] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Rusboost: Improving classification performance when training data is skewed. In *2008 19th Int. Conf. on Pattern Recognition*, pages 1–4. IEEE, 2008.

[107] J. P. Shaffer. Multiple hypothesis testing. *Annual review of psychology*, 46(1):561–584, 1995.

[108] S. K. Sharma, R. G. Amin, and S. Gattoufi. Choosing the best twenty20 cricket batsmen using ordered weighted averaging. *International Journal of Performance Analysis in Sport*, 12(3):614–628, 2012.

[109] M. Sokolova, N. Japkowicz, and S. Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint Conf. on artificial intell.*, pages 1015–1021. Springer, 2006.

[110] M. Sugeno. *Theory of fuzzy integral and its applications*. PhD thesis, Tokyo Institute of Technology, 1974.

[111] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, 1974.

[112] M. Sugeno. Fuzzy measures and fuzzy integrals—a survey. In *Readings in fuzzy sets for intelligent systems*, pages 251–257. Elsevier, 1993.

[113] H. Tahani and J. M. Keller. Information fusion in computer vision using the fuzzy integral. *IEEE Trans. on Systems, Man and Cybernetics*, 20(3):733–741, 1990.

[114] A. F. Tehrani, W. Cheng, K. Dembczyński, and E. Hüllermeier. Learning monotone nonlinear models using the choquet integral. *Machine Learning*, 89(1-2):183–211, 2012.

[115] A. F. Tehrani, W. Cheng, and E. Hüllermeier. Choquistic regression: Generalizing logistic regression using the choquet integral. In *Proc. of the 7th Conf. of the European Soc. for Fuzzy Logic and Tech.*, pages 868–875. Atlantis Press, 2011.

[116] A. F. Tehrani, W. Cheng, and E. Hüllermeier. Choquistic regression: Generalizing logistic regression using the choquet integral. In *Proc. EUSFLAT*, pages 868–875, 2011.

[117] A. F. Tehrani, W. Cheng, and E. Hullermeier. Preference learning using the choquet integral: The case of multipartite ranking. *IEEE Trans. on Fuzzy Systems*, 20(6):1102–1113, 2012.

[118] S. Tesfamariam, R. Sadiq, and H. Najjaran. Decision making under uncertainty—an example for seismic risk management. *Risk Analysis: An International Journal*, 30(1):78–94, 2010.

[119] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[120] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[121] A. N. Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

[122] A. Tsanas and A. Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

[123] A. D. Waegenaere and P. P. Wakker. Nonmonotonic Choquet integrals. *J. Mathematical Economics*, 36:45–60, 2001.

[124] C. Wagner and D. T. Anderson. Extracting meta-measures from data for fuzzy aggregation of crowd sourced information. In *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 1–8, Brisbane, Australia, 2012.

[125] X. Wang, A. Chen, and H. Feng. Upper integral network with extreme learning mechanism. *Neurocomputing*, 74(16):2520–2525, 2011.

[126] Z. Xu, R. Jin, H. Yang, I. King, and M. Lyu. Simple and efficient multiple kernel learning by group lasso. In *Proc. Int. Conf. Machine Learning*, pages 1175–1182, 2010.

[127] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[128] R. R. Yager. Owa aggregation over a continuous interval argument with applications to decision making. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):1952–1963, 2004.

[129] R. R. Yager and G. Beliakov. Owa operators in regression problems. *IEEE Transactions on Fuzzy Systems*, 18(1):106–113, 2009.

[130] R. R. Yager and D. P. Filev. Induced ordered weighted averaging operators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(2):141–150, 1999.

[131] I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[132] I.-C. Yeh and T.-K. Hsu. Building real estate valuation models with comparative approach through case-based reasoning. *Applied Soft Computing*, 65:260–271, 2018.

[133] I.-C. Yeh and T.-K. Hsu. Building real estate valuation models with comparative approach through case-based reasoning. *Applied Soft Computing*, 65:260–271, 2018.

[134] M. Yeheyis, K. Hewage, M. S. Alam, C. Eskicioglu, and R. Sadiq. An overview of construction and demolition waste management in canada: a lifecycle analysis approach to sustainability. *Clean Technologies and Environmental Policy*, 15(1):81–91, 2013.

[135] L. Yuan, J. Liu, and J. Ye. Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems*, pages 352–360, 2011.

[136] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[137] J. Zhai, H. Xu, and Y. Li. Fusion of extreme learning machine with fuzzy integral. *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 21(2):23–34, 2013.

Appendix A

Table A.1
Data sets used in the experimental evaluation

Data set name	n	d	**Details**	**Dependent variable**
Concrete [131]	1,030	8	Given the composition of concrete (cement, fly ash, fine and course aggregates, water, age, etc., regression problem is the predict the compressive strength.	Concrete compressive strength
Real Estate [133]	414	5	The market historical data set of real estate valuation are collected from Sindian Dist., New Taipei City, Taiwan.	House price per unit area
Fish Toxicity [23]	908	6	Data set containing values for 6 attributes (molecular descriptors) of 908 chemicals used to predict quantitative acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow)	Aquatic toxicity towards the fish (fathead minnow), measured in mol/L.
Aquatic Toxicity [22]	546	8	Data set containing values for 8 attributes (molecular descriptors) of 546 chemicals used to predict quantitative acute aquatic toxicity towards planktonic crustacean Daphnia Magna.	Aquatic toxicity towards planktonic crustacean Daphnia Magna, measured in mol/L.
Red Wine [31]	1599	11	A dataset of red vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests	Quality (score between 0 and 10)
White Wine [31]	4898	11	A dataset of white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests	Quality (score between 0 and 10)
ENB-2 [122]	768	8	Energy analysis using 12 different building shapes. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. Objective is to predict the Cooling load.	Two variables: Heating load and Cooling load. We randomly chose just the second variable for the regression model.
Yacht [44]	308	6	A data set to predict the hydodynamic performance of sailing yachts. Inputs include the basic hull dimensions and the boat velocity.	Residuary resistance per unit weight of displacement
Airfoil [19]	1503	5	NASA data set, obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel.	Scaled sound pressure level, in decibels.
ISE [1]	536	7	Data sets includes returns of Istanbul Stock Exchange with seven other international index; SP, DAX, FTSE, NIKKEI, BOVESPA, MSCE_EU, MSCI_EM from Jun 5, 2009 to Feb 22, 2011.	Istanbul stock exchange national 100 index

Table A.2
Data sets used in the experimental evaluation

Data set name	n	d	Details	Classification task
Vertebral [33]	310	6	Biomedical data set built by Dr. Henrique da Mota. Each patient is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (in this order): pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis.	To classify patients as belonging to one out of two categories: Normal (100 patients) or Abnormal (210 patients).
Biodeg [33]	1055	41	The QSAR biodegradation dataset was built in the Milano Chemometrics and QSAR Research Group. The objective is to study the relationships between chemical structure and biodegradation of molecules. Data comprises 41 molecular descriptors of 1055 chemicals.	To classify the instances into discriminate-ready (356) and not-ready (699) biodegradable molecules.
Mmass [33]	961	6	Data comprises the attributes from the mammography tests performed for breast cancer screening. This data set can be used to predict the severity (benign or malignant) of a mammographic mass lesion. These digital mammograms were collected at the Institute of Radiology of the University Erlangen-Nuremberg between 2003 and 2006.	To classify the instances into benign and malignant categories.
Breast Cancer [33]	286	9	This data set includes 201 instances of negative class and 85 instances of positive class. The instances are described by 9 attributes including age, tumor-size, menopause, and breast-quad	To classify as positive and negative cases of breast cancer.
Transfusion [33]	748	5	These data comprise blood donation related information on 748 donors, each instance included R (Recency - months since last donation), F (Frequency - total number of donations), M (Monetary - total blood donated in c.c.), T (Time - months since first donation)	To predict if he/she would donate blood in the next month (March 2007).
Heart [33]	270	13	A dataset of physical attributes and assessed medical parameters of people for the detection of the presence of heart disease.	To classify between people with and without heart disease.
Sonar [33]	208	60	The dataset contains 111 instances of sonar signals bouncing off a metal cylinder and 97 instances of signals bouncing off rocks under similar conditions. Each signal is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time.	To classify the objects as metal cylinders and rocks.
Dermatology [33]	366	34	The data comprises of patient information which include 12 clinically evaluated features, and 22 histopathological features determined by an analysis of skin samples under a microscope.	Detect the presence of the skin disease.
Ecoli [33]	336	8	This data contains protein localization sites in the Ecoli bacteria. Among the 8 classes omL, imL, and imS are the minority classes and used as outliers. All the other majority classes are used as inliers. Labels (1 = outliers, 0 = inliers)	To segregate the outliers and the inliers.
Glass [33]	214	10	This data relates to the study of classification of types of glass which was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence. Using the available chemical and physical properties of the glass, the objective is to predict if the glass was a type of "float" glass or not.	To classify between two types of glass material.
Wine [33]	178	13	These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars (3 classes). The analysis determined the quantities of 13 constituents found in each of the three types of wines. The classes 2 and 3 are merged to a single class. The objective is classify the origin of the wine as the first cultivar vs. the other two.	To identify cultivar associated with the wine.
Ionosphere [33]	351	34	This is radar data collected by a system in Goose Bay, Labrador. The system consists of a phased array of 16 high-frequency antennas. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.	To classify the instances as "Good" and "Bad" radar returns.
Absentee [33]	740	21	The database was created with records of absenteeism at work from July 2007 to July 2010 at a courier company in Brazil. The available attributes on each individual cover physical attributes, education and social attributes, health, and some miscellaneous attributes. The objective is to predict the absenteeism time in hours. We transformed this to a binary classification problem by assigning instances with less than 4 hours of absenteeism as the first class and the rest as the second class.	To segregate the instances into the two classes of absenteeism.
Abalone [33]	4177	8	Traditionally, the age of Abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a very time-consuming task. This data consists of the physical attributes like dimensions and weight of Abalone. The objective is to predict the number of rings (which can be used to calculated the age). We transformed this to a binary classification problem by merging the instances with less than 10 rings as the first class and the rest as the second class.	To segregate the instances into the two classes of Abalone.