Data-Driven Compression and Efficient Learning of the Choquet Integral

Muhammad Aminul Islam, *Student Member, IEEE,* Derek T. Anderson, *Senior Member, IEEE,* Anthony J. Pinar, *Member, IEEE,* Timothy C. Havens, *Senior Member, IEEE*

***Abstract*—The *Choquet integral* (ChI) is a parametric nonlinear aggregation function defined with respect to the *fuzzy measure* (FM). To date, application of the ChI has sadly been restricted to problems with relatively few numbers of inputs; primarily as the FM has 2^N variables for N inputs and $N(2^{N-1}-1)$ monotonicity constraints. In return, the community has turned to density-based imputation (e.g., Sugeno λ-FM) or the number of *interactions* (FM variables) are restricted (e.g., k-additivity). Herein, we propose a new scalable data-driven way to represent and learn the ChI, making learning computationally manageable for larger N. First, data supported variables are identified and used in optimization. Identification of these variables also allows us recognize future ill posed fusion scenarios; ChIs involving variable subsets not supported by data. Second, we outline an imputation function framework to address data unsupported variables. Third, we present a lossless way to compress redundant variables and associated monotonicity constraints. Last, we outline a lossy approximation method to further compress the ChI (if/when desired). Computational complexity analysis and experiments conducted on synthetic data sets with known FMs demonstrate the effectiveness and efficiency of the proposed theory.**

***Index Terms*—Data/information fusion, fuzzy integral, Choquet integral, fuzzy measure, data-driven learning**

I. Introduction

DATA/INFORMATION fusion is an enabling theory for numerous fields, e.g., machine learning, signal/image processing, big data, *Internet Of Things* (IoT), bioinformatics, and cyber security, to name a few. In this paper, we focus specifically on aggregation as the term *fusion* has been elusive definition-wise (either too vague or overly specific). In general, the idea is to combine N different inputs in such a way that the overall result (typically a reduction from N inputs to one result) is somehow better than the outcome acquired using just the individuals by themselves. First, it is up to the user to define "better". For example, maybe the idea is to combine a set of inputs to create a single result that can be more easily visualized. The idea could also be to reduce (summarize) data so it is more manageable. In machine learning, better may mean achieving more generalizable decision boundaries for classifiers. The point is, "better" is a concept that needs to be specified relative to some task at hand. Next, focus shifts to how to combine these N inputs. To date, most mathematical approaches have focused on combining inputs relative to the assumption of independence between them (which is advantageous tractability-wise). However, often there are rich interactions (e.g., correlations) between inputs that should be exploited. But for N inputs, there are 2^N possible subsets to consider. As N grows, tractability is of utmost concern. The focus of this paper is a new tractable way to identify, model and exploit non-redundant data supported interactions. The ideas are presented at an abstract level as to not muddle the theory with any one particular application.

Herein, we focus on the *fuzzy integral* (FI) as it is a powerful and flexible aggregation function capable of exploiting rich interactions between inputs. In 1972, Sugeno introduced the *fuzzy measure* (FM) (a normal capacity) in the context of the *Sugeno integral* (SI) [1]. Though Sugeno coined the term FI for his SI, the term FI has been generalized to a wider class of integrals. One well-known example is the *Choquet integral* (ChI), originally proposed by Gustav Choquet in 1953 [2]. While the ChI was initially used in statistical mechanics and potential theory, in particular with respect to an additive probability measure, it has since found application in numerous other areas, e.g., computer vision [3, 4], classification [5–10], pattern recognition [11–13], *multi-criteria decision making* (MCDM) [14–20], control theory, forensic science [21], Choquistic regression [22, 23], and *multi-kernel learning* (MKL) for *support vector machine* (SVM) classification and regression [24]. Numerous algorithms have been put forth to learn the FI from data, e.g., *quadratic programming* (QP) [25], gradient descent [26], penalty/reward [27], Gibbs sampler [28], and linear programming [29], to name a few.

For N inputs, there are 2^N FM variables and $N(2^{N-1}-1)$ monotonicity constraints. An advantage of the FI is we can model and exploit such knowledge. However, a drawback is lack of tractability. In practice, this is an important and often limiting factor. The community has been exploring ways to solve this dilemma. The traditional approach is to require or learn just the densities (measure on just the singletons) and an imputation function is used to *fill in* the remaining variables, e.g., Sugeno characteristic polynomial and resultant λ-measure [30]. In a different approach, Grabisch defined the k-order additive FM/FI [31]. The k-additive FM/FI is a restriction limiting interactions to at most k inputs. It can do this because the Mobius values for sets (variables) larger than k are zero. Exploiting this property enables us to discard FM variables and obtain a lossless compression. The k-order additive measure is indeed efficient when k is much less than the number of inputs. In many applications, e.g., MCDM, this often proves to be sufficient and of great utility. However, in other situations, e.g., pattern recognition, machine learning, signal/image processing, and computer vision, to name a few, we are often not dealing with humans per se versus automation and notions like bounded rationality do not apply. It can, and often is, the case that higher-order interactions exist and are

crucial. Last, even if we can determine the order k, many problems render k-additivity ineffective. For example, the heavily utilized minimum and maximum aggregation operators–and other *linear combinations of order statistics* (LCOS) at that–are of N-order, requiring all 2^N MT terms for the FM and FI computation. Consequently, there is no savings in the number of variables. On the contrary, the computational complexity increases significantly, as the computation of the FI based on the MT is highly dense (needs all 2^N terms) verses the conventional formulation of the FI (which needs only N terms). In closing, density-based imputation and k-order additivity have been explored to date and are applicable and of great use for different problems/contexts.

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

This work is driven by the fact that a single *instance* of the FI for N inputs uses only N of the 2^N FM variables. Hence, we can learn at most $\min\{2^N, M \times N\}$ variables for a problem with M observations and N inputs. We know from linear algebra that in order to obtain unique solutions for 2^{20} FM variables in a 20 input problem, we need at least 2^{20} independent observations, which is more than one million observations. While a problem with input sizes around 20 is common, it is rare to find this huge number of samples for that problem. Now suppose the problem has $1,000$ observations, then at most $(20 \times 1,000) = 20,000$ variables are needed, when each observation is associated with unique set of variables, to represent the FI for all the observations. In reality, the actual number of variables is far fewer because many observations share variables among them. Another significant advantage from using only the data-supported variables is that it reduces the monotonicity constraints substantially, which is exponential on the inputs for the full FI and, therefore, can be a limiting factor in many solvers. A sophisticated solver could possibly incorporate as many as $20,000$ variables, however, handling more than one million variables along with exponential order of constraints becomes near impossible for any kind of modern day solver and computing platform.

Last, before we delve into related work and new methods, an explanation of why we explore the QP for optimization is given. One of the most commonly encountered error functions in practice is the *sum of squared error* (SSE). This captures how different our learned target function is to a ground truth. However, other error and/or associated penalty functions can and have been used relative to learning aggregation operators. For example, in [32] Bustince et al. discussed problems related to definitions of penalty functions in the context of data aggregation. They gave examples of continuous penalty functions based on spread measures including standard deviation and variance and discussed the idea to define a penalty function for non-monotonic aggregation function. In [33], we investigated ℓ_p norm regularization to balance function error with minimum complexity FMs. The message is, herein we focus on a generic four step process for data-driven FI learning. The concepts put forth can be used by different solvers, e.g., particle swarm optimization, genetic algorithms, etc., based on a user's desired error and/or penalty function. Our focus is the four steps, not a particular solver.

The remainder of this paper is arranged as follows. Section II describes how to learn the FI, specifically the ChI, from data that includes the full set of FM variables. Section III is an example of a small problem to illustrate how the proposed method works. Section IV details the proposed new methods followed by experiments and analysis in Section V.

II. Background

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

(i) Boundary condition: $\mu(\emptyset) = 0$,
(ii) Monotonicity: if $A, B \subseteq X$ and $A \subseteq B$, $\mu(A) \leq \mu(B)$.

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

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

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

where π_j is a permutation function for observation $\mathbf{o}_j$ on the indices that satisfies $0 \leq o_j(x_{\pi(1)}) \leq \ldots \leq o_j(x_{\pi(N)})$, where $S_{\pi_j(i)} = \left\{x_{\pi_j(i)}, x_{\pi_j(i+1)}, \ldots, x_{\pi_j(N)}\right\}$ and $o_j(x_{\pi_j(0)}) = 0$ [34]. Eq. (1) can be written in matrix form as $C_\mu(\mathbf{o}_j) = \mathbf{c}_j^T \mathbf{u}$, where $\mathbf{c}_j$ is a column vector containing the $(2^N - 1)$ coefficients for observation $\mathbf{o}_j$. Let k be the index of variable $\mu(B \in 2^X)$ in $\mathbf{u}$. Then the kth element of $\mathbf{c}_j$ is $c_{jk} = o_j(x_{\pi_j(l)}) - o_j(x_{\pi_j(l-1)})$ if $\exists S_{\pi_j}(l) = B,\ l \in \{1, \ldots, N\}$, and 0 otherwise. The FM monotonicity constraints can be written as $\mu(A) \leq \mu(B), \forall A, B \subseteq X$ and $A \subseteq B$. The set of monotonicity constraints defined by the above

inequality relations are exhaustive; however, there are many redundant constraints among them which can be excluded for an optimization problem. For example, if we include $\mu(\{x_1\}) \leq \mu(\{x_1, x_2\})$ and $\mu(\{x_1, x_2\}) \leq \mu(\{x_1, x_2, x_3\})$ as monotonicity constraints, then they also imply that $\mu(\{x_1\}) \leq \mu(\{x_1, x_2, x_3\})$, and there is no need to explicitly define all the relations. The minimal set of constraints for an FM is $\mu(A) \leq \mu(A \cup q), \forall A \subset X$ and $\forall q \in X, q \notin A$.

The SSE between the ChI for all the observations in the training data, O, and our labels is [25, 35]

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

Based on this, the least square minimization problem can be expressed as [25, 35]

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

$$\mu(A) \leq \mu(A \cup q), \quad \forall A \subset X \text{ and } \forall q \in X, q \notin A, \quad \text{(monotonicity conditions)} \tag{3a}$$

$$\mu(\emptyset) = 0, \quad \text{(boundary conditions)} \tag{3b}$$

$$\mu(X) = 1, \quad \text{(normality conditions)} \tag{3c}$$

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

III. Example 1: Data Supported Variables

In this section, we provide a simple numeric example of the underlying principle of the proposed method for a simple three input case ($N = 3$). Figure (1) shows the true underlying FM. Table I is an example training data set, O, with 5 instances and labels, drawn randomly from the true underlying FM.

Example 1 has seven variables, denoted by $\mathbf{u}$, and five instances (training data). According to Eq. (1), the ChI for each instance requires FM variables for three sets, $S_{\pi_j(i)}$, $i = \{1, 2, 3\}$ (column four in Table I). In Example 1, only six of the seven variables, denoted as $\mathbf{u}_P$, are encountered–shown in column five of Table I. Let us denote the unused variable, $\mu(\{x_2, x_3\})$, as $\mathbf{u}_Q$. We can split the structure in Figure (1) into two based on the variables, $\mathbf{u}_P$ and $\mathbf{u}_Q$ (Figure (2)). The variables $\mathbf{u}_P$ can be learned by solving an optimization problem with only $\mathbf{u}_P$ as there are five unique ChI equations for five variables, and $\mu(X)$ is constant. On the other hand, there is no ChI equation involving $\mathbf{u}_Q$, and its value can be anywhere in the valid range, which is obtained using $\mathbf{u}_P$ and the monotonicity constraints on $\mathbf{u}_Q$. An imputation function, discussed in detail later, can be employed to assign a specific value within the interval range.

IV. Efficient ChI learning

A. Optimization with respect to just data supported variables

Based on our training data, we can partition the FM variables into two parts. The first set, $P \subseteq 2^X$, specifically

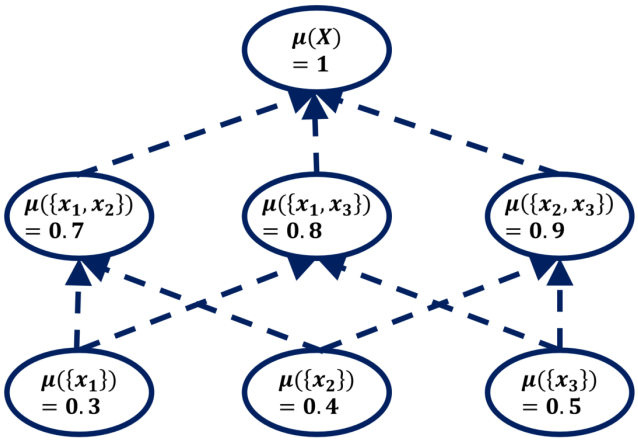


Fig. 1. **Example 1**. Illustration of known (aka reference) FM for three inputs ($N = 3$). Nodes are variables and edges are monotonicity constraints.

$P = \{S_{\pi_j(i)}\}, \forall i \in \{1, 2, \ldots, N\}$ and $\forall j \in \{1, 2, \ldots, M\}$, is all variables that appear at least once in the ChI formula with respect to O ($\mathbf{o}_i, i = 1, \ldots, M$) and the second set, $Q = 2^X \setminus P$, is all other variables. Let the cardinality of P and Q be p and q respectively. The partitioning of the power set leads to the decomposition of the vector $\mathbf{u}$; $\mathbf{u} = [\mathbf{u}_P \ \mathbf{u}_Q]$, where $\mathbf{u}_P(l) = \mu(A), \ A \in P, \ l = \{1, 2 \ldots, p\}$, and $\mathbf{u}_Q(k) = \mu(B), \ B \in Q, \ k = \{1, 2 \ldots, q\}$. The coefficient vector $\mathbf{c}_j$ for each observation $\mathbf{o}_j$ is $\mathbf{c}_j = [\mathbf{c}_{Pj} \ \mathbf{c}_{Qj}]^T$, where $\mathbf{c}_{Pj}$ and $\mathbf{c}_{Qj}$ are the respective coefficient vectors of $\mathbf{u}_P$ and $\mathbf{u}_Q$ for the given observation. As the variables $\mathbf{u}_Q$ are not present in the ChI definitions of all the observations, their coefficients in the quadratic formula with respect to the training data are always zeros, i.e., $\mathbf{c}_{Qj} = \mathbf{0}, \ \forall j \in \{1, 2, \ldots, M\}$.

The objective function in OP1 consists of a quadratic term with a coefficient matrix H and a linear term with coefficient vector $\mathbf{d}$. These can be represented in terms of the observation coefficients, and thus can be partitioned into *blocks*. The coefficient matrix, H, is therefore

$$\begin{aligned} H = \sum_{j=1}^{M} \mathbf{c}_j \mathbf{c}_j^T &= \sum_{j=1}^{M} \begin{bmatrix} \mathbf{c}_{Pj} \\ \mathbf{c}_{Qj} \end{bmatrix} [\mathbf{c}_{Pj}^T \quad \mathbf{c}_{Qj}^T] \\ &= \sum_{j=1}^{m} \begin{bmatrix} \mathbf{c}_{Pj} \mathbf{c}_{Pj}^T & \mathbf{c}_{Pj} \mathbf{c}_{Qj}^T \\ \mathbf{c}_{Qj} \mathbf{c}_{Pj}^T & \mathbf{c}_{Qj} \mathbf{c}_{Qj}^T \end{bmatrix} \\ &= \begin{bmatrix} 2 \sum_{j=1}^{M} \mathbf{c}_{Pj} \mathbf{c}_{Pj}^T & \mathbf{0}_{PQ} \\ \mathbf{0}_{QP} & \mathbf{0}_{QQ} \end{bmatrix} \\ &= \begin{bmatrix} H_{PP} & \mathbf{0}_{PQ} \\ \mathbf{0}_{QP} & \mathbf{0}_{QQ} \end{bmatrix}, \end{aligned} \tag{4}$$

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

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

where $\mathbf{0}_Q$ is a $Q \times 1$ vector of all zeros and $\mathbf{d}_P = -2 \sum_{j=1}^{M} y_j \mathbf{c}_{Pj}$. We can see from the alternate representation

TABLE I
EXAMPLE 1: TRAINING DATA-SET FOR A THREE INPUT CASE ($N = 3$).

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

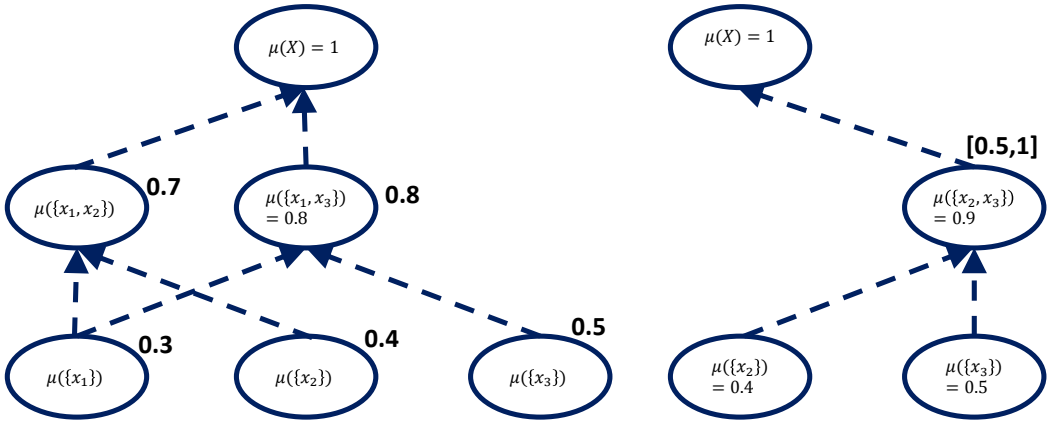


Fig. 2. **Example 1**. (a) Illustration of required FM values for data in Table I. Note, $\mu(\{x_2, x_3\})$ is not supported by training data and subsequently cannot be learned. (b) Illustration of data unsupported values and their interval-valued ranges due to monotonicity conditions. The values/intervals outside nodes signify that they are learned via optimization whereas those inside are used as constants.

of the coefficient matrix H in Eq. (4) that the diagonal blocks of the matrix are zeros. This indicates that variables $\mathbf{u}_P$ and $\mathbf{u}_Q$ in the quadratic terms are decoupled, and consequently the objective function can be represented as a linear sum of two functions with variables $\mathbf{u}_P$ and $\mathbf{u}_Q$. That is, $f_O(\mathbf{u}) = f_O(\mathbf{u}_P, \mathbf{u}_Q) = f_{O_1}(\mathbf{u}_P) + f_{O_2}(\mathbf{u}_Q)$.

Furthermore, the constraints can be divided into groups with respect to $\mathbf{u}_P$ and $\mathbf{u}_Q$. For sets $A, B \in 2^X$, we have the following four cases: (1) $\mu(A), \mu(B) \in \mathbf{u}_P$, (2) $\mu(A), \mu(B) \in \mathbf{u}_Q$, (3) $\mu(A) \in \mathbf{u}_P, \mu(B) \in \mathbf{u}_Q$, and (4) $\mu(A) \in \mathbf{u}_Q, \mu(B) \in \mathbf{u}_P$. The boundary constraints can also be grouped based on: (1) $\mu(A) \in \mathbf{u}_P$ and (2) $\mu(A) \in \mathbf{u}_Q$. Now, we rewrite the optimization problem in OP1 in terms of $\mathbf{u}_P$ and $\mathbf{u}_Q$,

$$\begin{aligned}(\text{OP2}) \min_{\mathbf{u}_P,\mathbf{u}_Q} \; & f_O(\mathbf{u}_P,\mathbf{u}_Q) = \mathbf{u}_P^T H_{PP}\mathbf{u}_P + \mathbf{u}_P^T\mathbf{0}_{PQ}\mathbf{u}_Q + \\ & \mathbf{u}_Q^T\mathbf{0}_{QP}\mathbf{u}_P + \mathbf{u}_Q^T\mathbf{0}_{QQ}\mathbf{u}_Q + \mathbf{d_P}^T\mathbf{u}_P + \mathbf{0}_Q\mathbf{u}_Q, \\ & = \underbrace{(\mathbf{u}_P^T H_{PP}\mathbf{u}_P + \mathbf{d_P}^T\mathbf{u}_P)}_{\geq 0,\text{ terms with only } u_P} + \\ & \underbrace{\mathbf{u}_P^T\mathbf{0}_{PQ}\mathbf{u}_Q + \mathbf{u}_Q^T\mathbf{0}_{QP}\mathbf{u}_P}_{=0,\text{ terms with both } u_P \text{ and } u_Q} + \underbrace{\mathbf{u}_Q^T\mathbf{0}_{QQ}\mathbf{u}_Q + \mathbf{0}_Q\mathbf{u}_Q}_{=0,\text{ terms with only } u_Q}, \end{aligned} \quad (5)$$

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

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

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

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

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

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

$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A), \mu(B) \in \mathbf{u}_Q, \quad (6a)$$
$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A) \in \mathbf{u}_P, \mu(B) \in \mathbf{u}_Q, \quad (6b)$$
$$\mu(A) \leq \mu(B) \text{ for } A \subset B \text{ and } \mu(A) \in \mathbf{u}_Q, \mu(B) \in \mathbf{u}_P, \quad (6c)$$
$$\mu(A) \geq 0 \text{ for } \mu(A) \in \mathbf{u}_Q, \quad (6d)$$

where $A, B \in 2^X$. It is worthwhile to note that OP2 includes the exhaustive set of monotonicity constraints and extended

list of boundary constraints only to facilitate our partitioning of the inequality constraints so we can decompose OP1. Instead of enumerating all possible monotonicity conditions, we instead define the monotonicity constraints, e.g., Eq. (3a) in the standard QP formulation, with the minimal set of relations excluding all redundant constraints. In the same manner, the boundary conditions can be scaled down, reducing the number of constraints considerably in both OP2.1 and OP2.2.

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

$$\text{(OP2.2a) find } \mathbf{u}_Q \text{ subject to } \mathrm{MC}(\mathbf{u}_Q) \tag{7}$$

where $\mathrm{MC}(\mathbf{u}_Q)$ denotes the constraints in Eqs. (6a-d). The valid region defined by these constraints is a convex bounded polyhedron in a q-dimensional space denoted by C_Q, where q is the cardinality of $\mathbf{u}_Q$. Any point inside C_Q is valid; therefore the whole convex polyhedron constitutes the solution set of the problem. Obviously, the problem has infinitely many solutions.

Our constraints can be further decomposed. Group I is unary constraints (Eqs. (6b-d)) since they include constants and variables from $\mathbf{u}_P$, which are themselves constants. In Group II, the constraints are binary with $\mathbf{u}_Q$ variables on both sides of the inequalities (Eq. (6a)). First, we consider the case of only Group I constraints. As these constraints specify that a variable is either less or greater than some constant, the hyper-lines for these constraints run parallel to the axes of a q-dimensional space. The resultant solution set for $\mathbf{u}_Q$, $\tilde{C}$, is therefore a hyper-rectangle with 2^q vertices. Alternately, we can say that the valid range of each variable with regard to Group I constraints lies in an interval. Let $I(x) = [i_l(x), i_u(x)]$ be the interval for variable $x = \mu(A) \in \mathbf{u}_Q$, which can be deduced from Group I constraints as $i_l(x) = \max_{(B \subset A, \mu(B) \in \mathbf{u}_P)} \mu(B)$ and $i_u(x) = \min_{(B \supset A, \mu(B) \in \mathbf{u}_P)} \mu(B)$. It is trivial to show that if $\mu(E), \mu(F) \in \mathbf{u}_Q$, and $E \subset F$, then $i_l(\mu(E)) \leq i_l(\mu(F))$ and $i_u(\mu(E)) \leq i_u(\mu(F))$ because each subset of E is also a subset of F and each superset of F is also a superset of E. Substituting the unary constraints with intervals, OP2.2a can be rewritten as; find $\mathbf{u}_Q$ subject to (i) $\mu(A) \leq \mu(B)$ for $A \subset B$ and $\mu(A), \mu(B) \in \mathbf{u}_Q$ and (ii) $i_l(\mu(A)) \leq \mu(A) \leq i_u(\mu(A)), \mu(A) \in \mathbf{u}_Q$. The solution set, C_Q, is a subset of $\tilde{C}$ constrained by (i), which means that a valid point within C_Q can be obtained from intervals by satisfying the monotonicity constraints on $\mathbf{u}_Q$.

In Section IV-C, we put forth an approach that allows data-unsupported variables to be computed on demand; hence we do not need to store or identify the valid region explicitly. That approach is based on the following concept. If $\forall A, B \in Q$, if $A \subset B$ then the interval calculated using Group I constraints always yields $I(A) \leq I(B)$ or $i_l(A) \leq i_l(B)$ and $i_u(A) \leq i_u(B)$. While $\tilde{C}$ can have a region over which $\mu(A) > \mu(B)$, the solution set, C_Q, does not contain any such region. Therefore, we can formulate a function, f_I, that maps an interval I in $[0, 1]$ to a point in the interval, i.e., $f_I(I) \in I$, such that for any $I_i < I_j$, f_I always preserves the relation $f_I(I_i) \leq f_I(I_j)$.

B. Computational Complexity

Here, we provide the computational complexity of the QP learning of data-supported variables. With training data of M samples for an N inputs problem, each observation can add at most N variables, and the total number of training variables in the efficient ChI, n_E, can reach to $2^N - 1$ variables at maximum for an overdetermined system. On the other hand, the number of training variables, n_S, in conventional ChI always is 2^{N-1} regardless of the training sample size. That is, $n_E \leq \min\{M \times N, 2^N - 1\}$ and $n_S = 2^{N-1}$. In a scenario when $M \times N < 2^{N-1}$, the worst case value for n_E is, $n_E(\text{worst}) = M \times N < 2^{N-1} = n_S$. This implies that the worst case time complexity of the efficient ChI is less than the standard ChI when $M \times N < 2^N - 1$ (or $M < \frac{2^N-1}{N}$) and is equal when $M \geq \frac{2^N-1}{N}$, at which point the two methods converges to exactly the same problem with 2^{N-1} variables. In reality, the number of observations for large N does not vary exponentially with N, but rather polynomially. For instance, we rarely encounter a 20 input problem with 2^{20} observations, but we face problems whose number of observations can be modeled as $(c \cdot 20^x)$, where c and x are arbitrary constants. As the time complexity of a convex QP is on the order of $O(k^3)$ for k variables, our complexity is $O((MN)^3) = O(N^{3(x+1)})$, where $M = N^x$. That is, we have polynomial time complexity under the assumption that the number of observations is polynomial with respect to N.

C. Imputation of data unsupported variables

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

Remark 1. Should I impute? In all data-driven ChI learning work that we are aware of, optimization is with respect to data supported and unsupported variables. However, what is really being assigned to those data unsupported variables and should we "trust" future decisions (fusions) that rely on one or more of these data unsupported variables? Unlike prior work, this paper informs the reader about how to identify such ill-posed fusion scenarios. Beyond that, it is ultimately up to the user/system to decide what to do. One strategy is to not fuse. Another is to fuse but report that data unsupported variables were used. Herein, we explore the decision of fusing using data supported variables and a philosophy that lets us control unsupported variable value assignment, versus making it arbitrary or a side effect of the optimization algorithm.

Remark 2. How is data-driven imputation different from density-driven imputation? Technically, imputation is the mapping of interval-valued uncertainty to scalars for variables unsupported by data (or densities respectively). By definition, we are not privileged to know the "true" answer to data (or density) unsupported variables. This is the reality and

type of problem with which we aim to advance. In density-based imputation, there are $2^N - 2 - N$ free (or *density unsupported*) variables. These variables have interval-valued uncertainty and a philosophy such as Sugeno's characteristic polynomial or a S-Decomposable measure is required for scalar assignment. In the context of our current paper, we are privileged to know (from data) much more than just the densities. Each observation *highlights* N variables of increasing cardinality ($\mu(\{x_{\pi_j(1)}\}), \mu(\{x_{\pi_j(1)}, x_{\pi_j(2)}\}), ...$). Relative to density-driven, data-driven imputation reduces the number of and narrows the interval widths of data unsupported variables.

Remark 3. How does data-driven imputation relate to k-additivity? In k-additivity, a subset of variables–those whose cardinality is greater than k–are, or are forced to be, *irrelevant* relative to the problem at hand. What this means is no imputation occurs. However, in our data-driven approach variables are selected at each level of cardinality. No assumption is made with respect to if a problem is k-additive or not. Therefore, imputation is needed to fill in that which we do not know (that which is not observable). It is important to note that k-additivity and data-driven imputation are not "in competition". They are different tools. If a problem is k-additive and the desire is to learn it from data, then our four step approach can be combined with k-additivity.

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

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

$$f_{w_I}(I) = (1 - w_I)i_l + w_I i_u, \text{ s.t., } w_I \in [0, 1] \tag{8}$$

In the Appendix, we prove that Eq. (8) satisfies the conditions of an imputation function for a monotonic w_I w.r.t I.

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

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

$$w_I(z_I(I)) = \frac{1}{1 + e^{-a(z_I(I)-b)}}, \tag{9}$$

where a and b are user/system parameters.

(Approach 2) Machine learning: Last, we discuss how f_I can be learned from data by fitting the function to the learned variable, $\mathbf{u}_P$. There are at least two ways to accomplish this; (i) solve an optimization problem based on a criteria like the SSE relative to the functions parameters or (ii) use a Hermite interpolation method to model this data with a monotonic piece-wise polynomial function [36]. The parameters of a dynamic weight function, w_I, can be learned from $\mathbf{u}_P$ following the below steps: First, determine the interval $I(x)$ for each variable, $x \in \mathbf{u}_P$ as if its actual value is unknown. That is, treat x as a variable while others as constants with known values and use the monotonicity constraints along with values of $\mathbf{u}_P \setminus x$ to obtain the interval. Next, calculate z_I according to $z_I(I(x)) = (1 - \beta)i_l(x) + \beta i_u(x)$ relative to a fixed β. Then, compute w_I (Eq. (8)) with the value of z_I calculated in the previous step and $f_I(I)$ as the actual (learned) FM value for variable $x \in \mathbf{u}_P$. Last, fit a monotonic function of w_I (e.g., sigmoid function in Eq. (9)) or piecewise monotonic polynomial function on the data z_I vs. $w_I(I)$ to estimate its parameter. As the weight in the fixed imputation function can be modeled as $w_I(z_I) = c$, therefore, it can also be learned in the same manner described above. Note, bisection method can be used to select an appropriated value for β in z_I equation, however, a fixed value of 0.5 was used in our experiments.

D. Lossless and lossy FM compression (variable elimination)

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

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

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

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

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

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

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

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

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

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

Min aggregation: The FM values for the min aggregation operator for an N-input system are $\mu(A) = 0, \forall A$ where $A \subset X$, and $\mu(X) = 1$, where V includes all the sets in the power set except X. V_U includes only those sets with cardinality $N-1$, $V_U = \{B\}, \forall |B| = N-1$, and V_L is empty. Therefore, $V_M = V \setminus (V_L \cup V_U)$, $|V_M| = 2^N - N - 1$, and in total $2^N - N - 1$ variables can be removed. Figure 3(a) shows, as an example, the full set of FM variables for a 3-input system on the left side and the compressed FM variables on the right. Only four variables, for V_L and X, are required for any imputation function to recover the remaining three. However, irrespective of problem size, the min imputation function needs only one variable, $\mu(X)$, which is by definition a constant.

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

Binary FM: An example of an arbitrary FM is shown in Figure 3(c). Of eight variables, four are 0-values and rest are 1-valued. Based on their values, these variables are partitioned into two clusters, V_1 and V_2, where V_1 contains 0-valued variables and V_2 has 1-valued variables. Using any imputation function, four variables can be removed, two from each cluster.

The above examples demonstrate the fact that we can reduce the number of variables significantly in a lossless way when a group of variables share the same value. In application, due to finite floating precision in computation, or simply to further reduce the number of required variables, we might want to incorporate a tolerance (ϵ) while removing variables, i.e.,

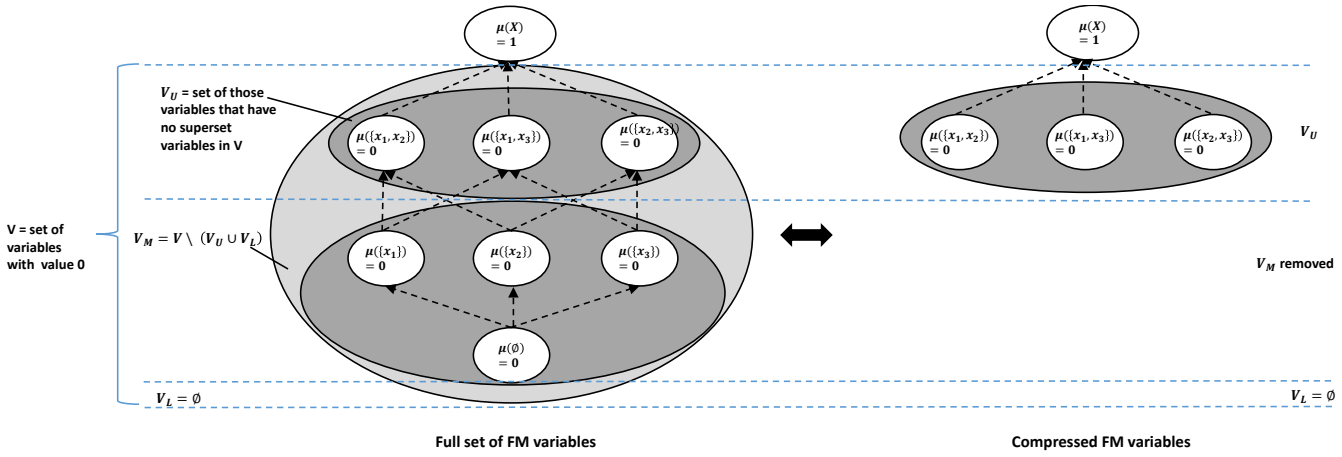
$$\max(\mu(v)) - \min(\mu(v)) \leq \epsilon, \forall v \in V, \text{ and } \epsilon > 0. \quad (10)$$

Note, Equation (10) is a lossy operation that yields ChI error.

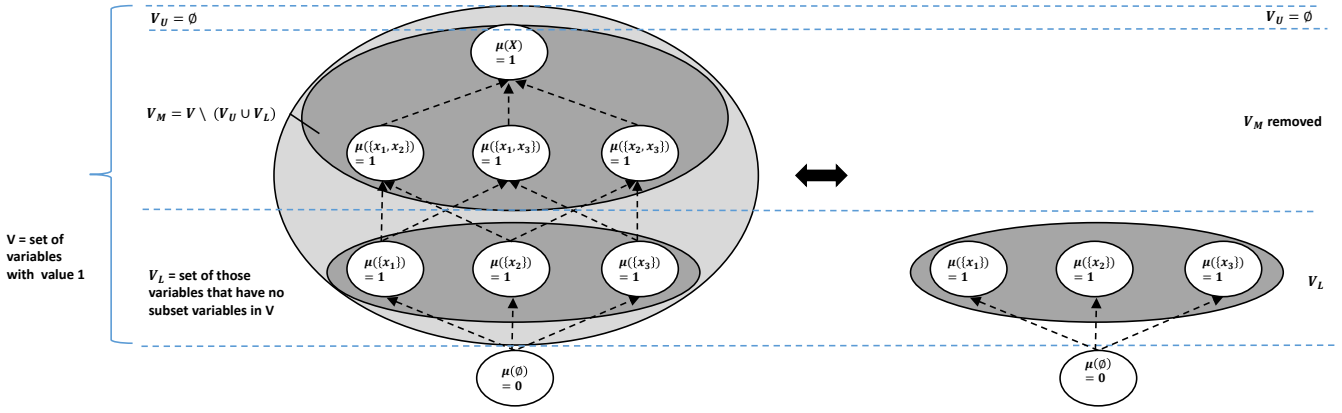
V. Experiments

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

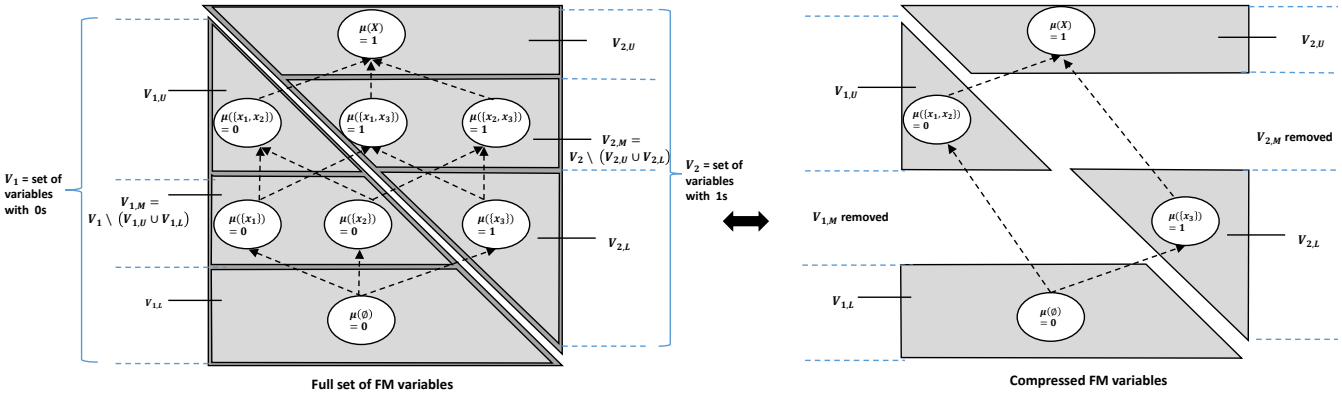
Data herein is generated (pseudo-)randomly from a uniform distribution. In experiments 1, 2 and 3, $N = 10$ is used, which yields $1,023$ FM variables and $5,110$ monotonicity constraints. The reason for $N = 10$ is because it is tractable (computationally and memory storage wise) by most solvers on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



(a)



(b)



(c)

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

Three OWAs are used to generate the labels for the data; soft-max, mean and soft-min (OWA weights provided in Table II). These OWAs were selected as they sample the "aggregation spectrum"—an optimistic operator, pessimistic operator and the last is an expected value operator. Furthermore, these specific operators highlight interesting scenarios relative to k-additivity.

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

In Experiment 1, we investigate the impact of using all FM variables versus just data supported FM variables. This allows us to explore two concepts. First, how much of a reduction are we really talking about in terms of numbers of variables? Second, experimentally, how much *variation* are we talking about with respect to imputation function selection versus learning with respect to all variables? Figure (4) shows the average number of training variables encountered and the MSE for soft-max, mean and soft-min. First, when there is sufficient data, e.g., $3,000$ training samples for $N = 10$, the standard and efficient ChIs (rightfully so) converge. Second, imputation enhances performance for undersampled problems when an appropriate function is selected. Third, the error amounts are low relative to our noteworthy savings in number of variables. It is important to highlight that while Experiment 1 shows that it is possible to achieve a good reduction in the number of FM variables, and associated monotonicity constraints–which leads to much needed improvements in optimization time–the exact amount of savings depends on data volume and variety. This is what we should expect since our method is a way to focus on efficiency relative to data supported variables and accountability with respect to what we do not know (imputation function).

B. Experiment 2: k-additivity

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

Figure (5) tells the following story. For situations where k is relatively (with respect to N) small and a good *fit*, e.g., Figure (5)(c) (the 1-additive mean operator), k-additivity does a wonderful job. However, when k is large, e.g., Figure (5)(b) and Figure (5)(d) (soft max and soft min), our approach does particularly well with respect to both MSE and the number of required variables. Last, at three thousand samples both the 10-additive and our data-driven ChI correspond (turn into) the full ChI. Note, the 10-additive ChI has slightly higher MSE—which is very low for both, 1.5×10^{-6} for the efficient Chi and 3.8×10^{-7} for the k-additive. The reason is due to dense computation of the ChI in k-additivity–which can have as many as $2^N - 1$ non-zero terms for N inputs when full-additive is used versus at most N non-zero terms for conventional ChI–the symmetric matrix D in QP optimization problem can be much denser relative to the standard/efficient ChI. Moreover, constraints in k-additivity involves more non-zero terms than the standard/efficient ChI and includes a global boundary constraint that contains all variables, the sum of which must be one. On the other hand, the standard/efficient ChI contains only unary and binary constraints.

C. Experiment 3: Imputation function exploration

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

It is intuitive that the quality of the results (see Figure (6)) depends on how closely the imputation function matches the true underlying FM. We see that the max and mean have lower MSE than min for soft-max. Similarly, the mean imputation function is the best for the mean FM, and the min imputation function has better results for the soft-min FM. The fixed-learned imputation function performs the best for all cases as it can capture the trend exemplified by the variables. For example, the learned imputation function for the soft-max FM was a soft-max, which could not be obtained using our included set of fixed imputation functions. The dynamic-learned imputation functions perform better than fixed but somewhat lags behind fixed-learned in two cases as the sigmoid function cannot exactly model a fixed weight function–which is the case for all three FMs. Using a more flexible function that can represent static functions could have better modeled the OWAs; however, there might be some cases for which sigmoid performs better than a fixed imputation function. Overall, a trade-off has to be made between simplicity and complexity while selecting an imputation function. Note, while using different imputation functions can definitely impact the efficient ChI's performance for relatively small sample sizes, their effects diminish as sample size increases because the number of variables needed to be imputed gradually decreases.

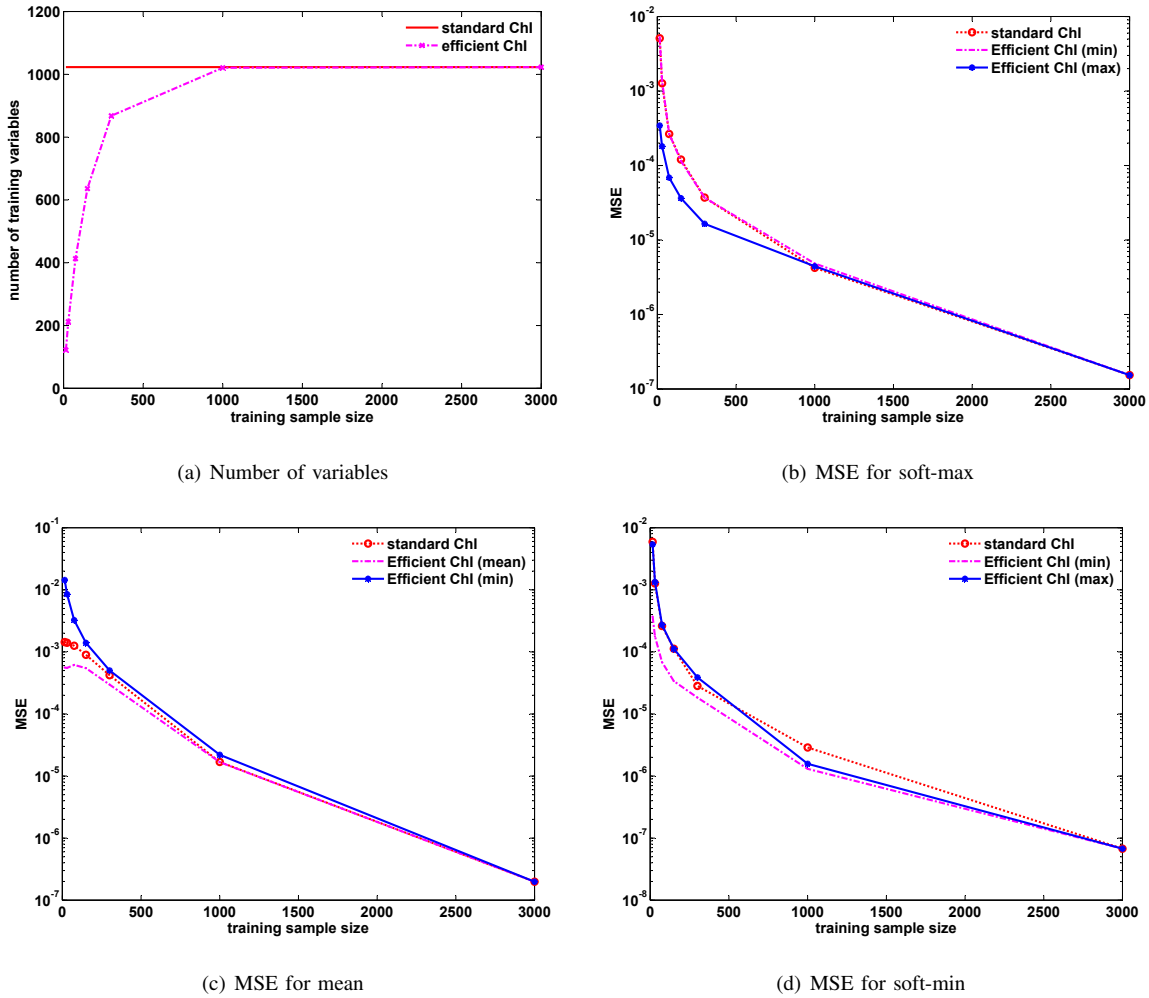
D. Experiment 4: (Relatively) Large N

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

The data set for Experiment 4 is (pseudo-)randomly generated from a uniform distribution. As described in Section IV-D,

TABLE II
OWA WEIGHTS USED IN EXPERIMENTS.

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



(a) Number of variables

(b) MSE for soft-max

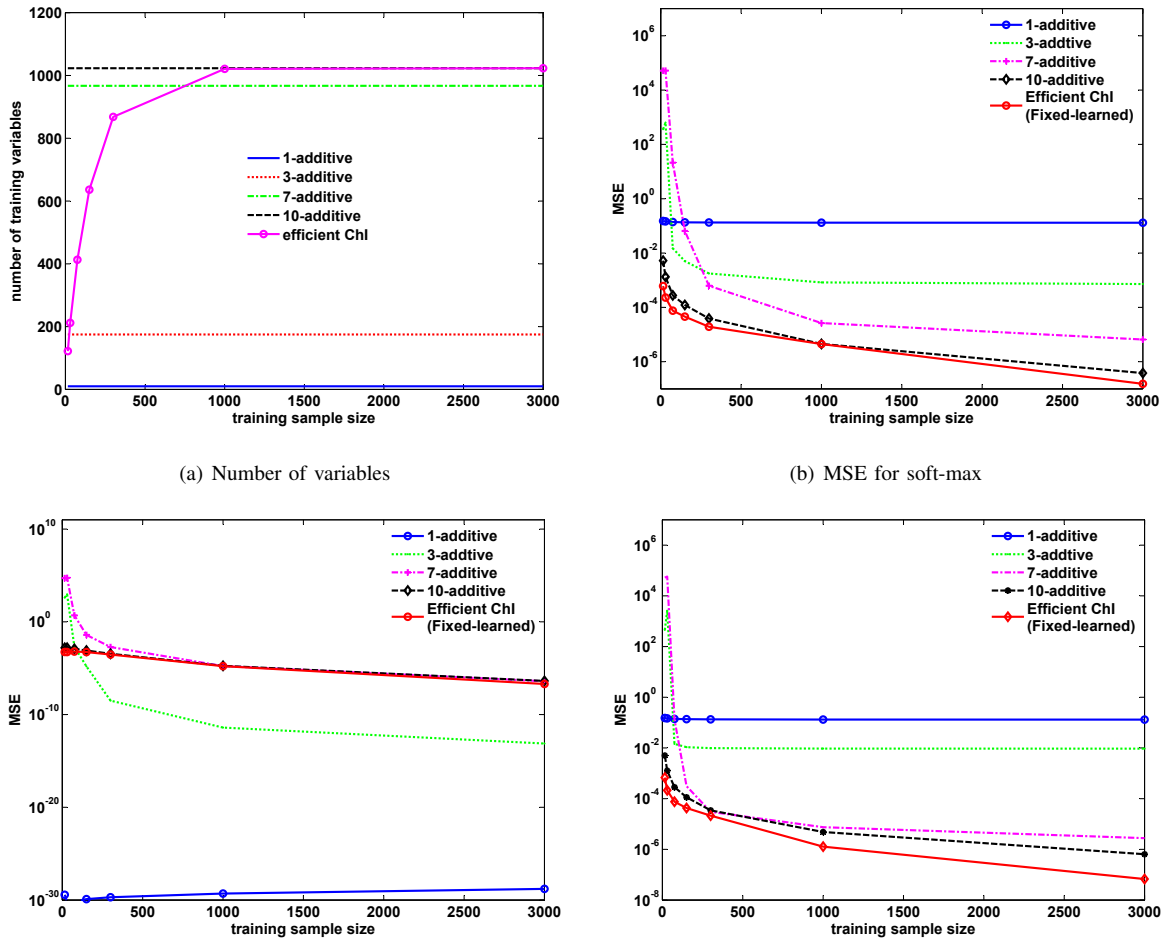
(c) MSE for mean

(d) MSE for soft-min

Fig. 4. Experiment 1. Comparison between the standard and efficient computation of the ChI.

we perform variable compression to investigate the impact on the SSE and the number of subsequent variables. We consider two values for tolerances, $\epsilon = 0.01$ and 0.001, which are used as thresholds for removing variables. For simplicity, instead of clustering variables with similar values, we checked the criteria defined in Eq. (10) for each individual variable and we removed it from the stored variables list if the criteria was satisfied. Figure (7) shows the results, where (a), (c), and (f) illustrate the impact that the imputation functions and tolerances have on the MSE whereas (b), (d), and (f) depict the impact on the number of variables. As expected, max imputation has lower error than mean for soft-max (Figure 7(a)), mean imputation function yields better results for mean (Figure 7(c)), and min imputation gives superior performance than mean for soft-min (Figure 7(e)).

As Figure (7) shows, imputation function selection impacts the MSE. For example, using max imputation has approximately four times improvement over mean for $1,000$ samples. On the contrary, changing the tolerances—with respect to the range under investigation—shows little-to-no impact, e.g., $\epsilon = 0.001$ provides the same results as no compression. While $\epsilon = 0.01$ has slightly worse MSE at lower training samples, it shows improved performance for relatively higher sample sizes (e.g., $1,000$). This may be because large numbers



(a) Number of variables

(b) MSE for soft-max

(c) MSE for mean

(d) MSE for soft-min

Fig. 5. Experiment 2: Joint exploration of k-additivity and our data-driven ChI method.

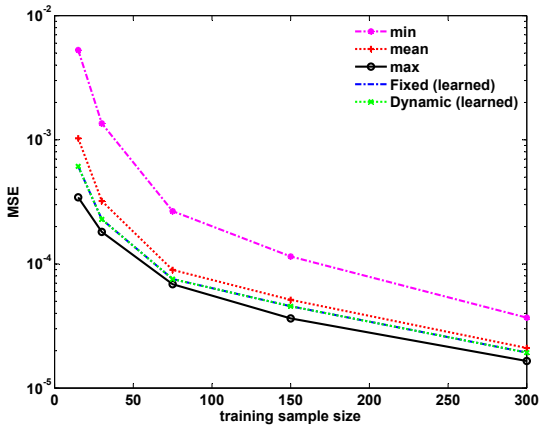
of samples generate huge numbers of variables, therefore removing relatively larger numbers of variables with closely related values and then imputing them with the same values can improve performance for the given soft-max and soft-min. Note, the soft-max has values close to one for variables residing at the upper layers in the lattice whereas soft-min has values close to zero at the bottom layers. This specific phenomena is unique to these types of FMs.

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

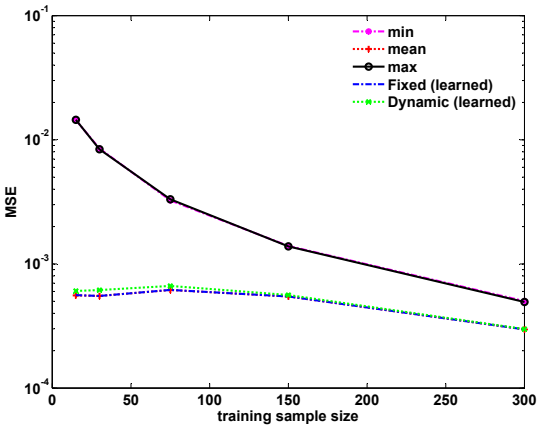
Last, Experiment 4 tells us that the MSE can reach an impressive rate (on the order of 10^{-3}) at a relatively small sample/variable size. This "cut-off point" depends on the acceptable error rate as well as the complexity of the underlying FM. The results suggest that when we have limited resources,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

VI. Conclusion

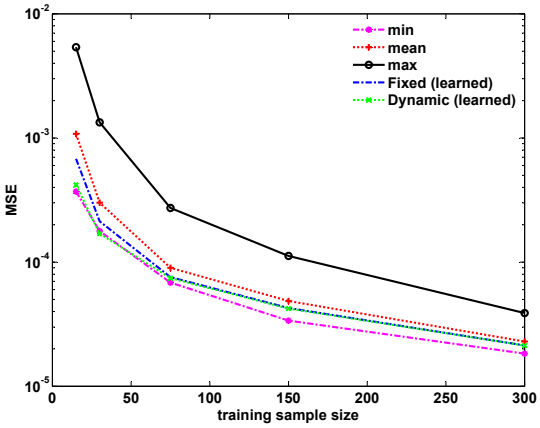
Herein, we introduced a new efficient and flexible data-driven way to represent and learn the FI. Specifically, variables supported by data are identified and used in optimization. Identification of data supported variables also allows us predict future ill posed input scenarios; ChI aggregations involving variable subsets that could not be inferred from data. Second, we outlined an imputation function framework for attacking data unsupported variables. Third, we presented a lossless



(a) MSE for soft-max



(b) MSE for mean

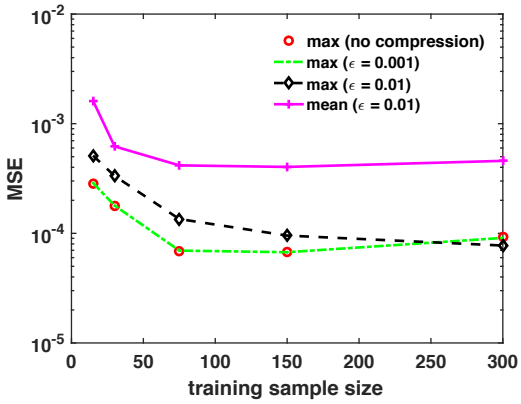


(c) MSE for soft-min

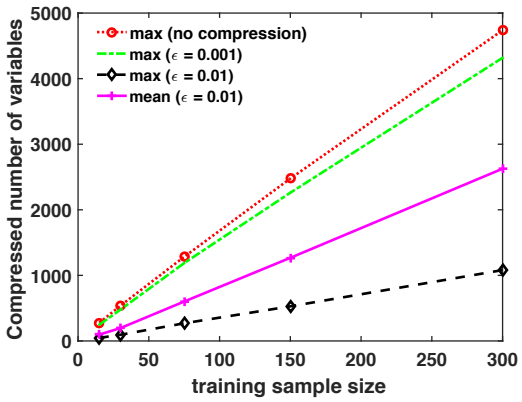
Fig. 6. Experiment 3: Results relative to different imputation functions.

method to compress redundant variables and monotonicity constraints. Last, we outlined a lossy approximation method to further compress the ChI (if/when desired). Optimization was cast in the context of quadratic programming. However, the four step process is independent of the underlying solver. Complexity analysis was provided and experiments were provided to allow more functional insight into the proposed theories. Advantages were shown relative to experiments in using all versus just data supported variables, k-additivity, impact of imputation function selection and computing the ChI for (relatively) large N. We noted that, like the k-additive integral and density-driven imputation, different applications have different needs and different amounts of knowledge are available. Overall, there does not seem to be a "best approach" but instead a set of tools for different problems/contexts.

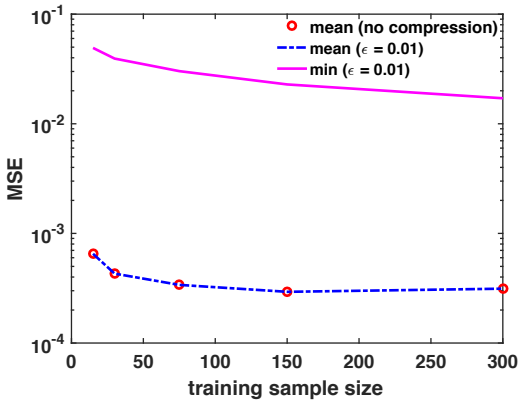
In future work, we will focus on improving the imputation function. For example, we will try to develop intuition to help a user map attributes of their problem to a particular function (or family of functions). Furthermore, we focused on solutions that produced a real-valued number from interval-valued information. However, we have extensions for the FI, both integrand and FM, that allow for computation with respect to interval and set-valued information. It will be interesting to instead of forcing a real-valued number to instead compute with the full uncertainty and discover what benefits, if any, there are in such an approach. We also plan to study how the use of an imputation function helps address overfitting. Furthermore, we partitioned the problem into data supported and data unsupported variables. However, within the set of data supported, not all variables are supported the same (number of samples). We would like to further study how to combat different degrees of "data supported" variables in learning. Next, we plan to explore different error and/or penalty functions and associated optimization algorithms. Since we learn from data and since the ChI has potentially many variables, over fitting can occur. The reader can refer to [33] for our prior work on regularization based learning of the ChI, which fits in naturally to the proposed paper and solver. However, it is likely that selection of imputation function can help as well. In future work, we will investigate questions like this to make the proposed work more generalizable. Last, the k-additive FI is a well-known and utilized approach in areas like MCDM, due to reasons like bounded rationality. In situations where it is desired and appropriate to learn a k-additivity solution, we



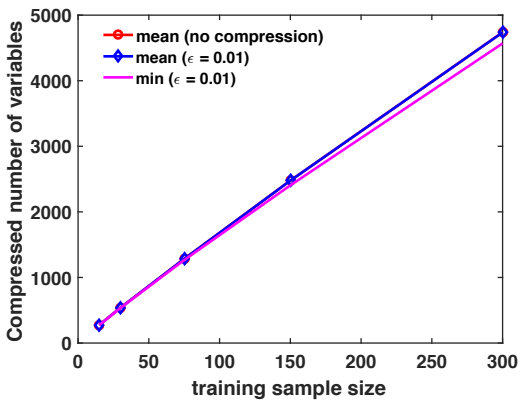
(a) MSE for soft-max



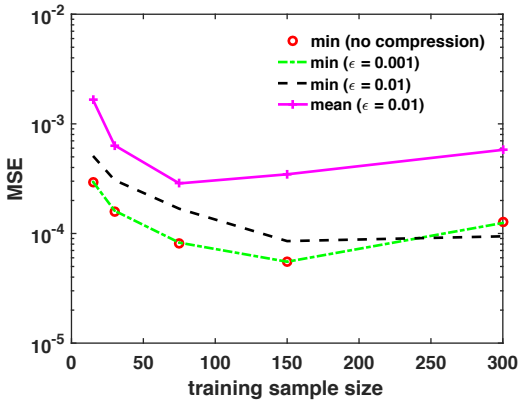
(b) Number of variables for soft-max



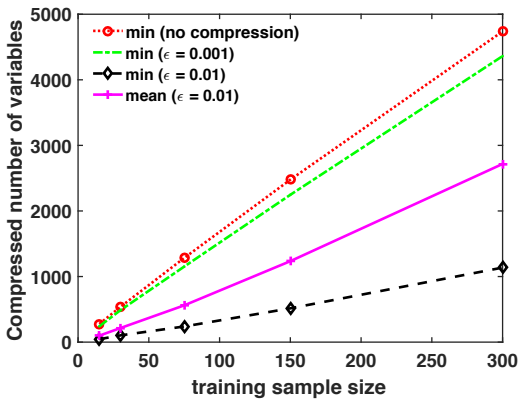
(c) MSE for mean



(d) Number of variables for mean



(e) MSE for soft-min



(f) Number of variables for soft-min

Fig. 7. Experiment 4: Efficient ChI results for $N = 20$ with and without variable compression for different tolerances.

will explore the impact of integrating our proposed four step data-driven learning to realize an improved technique; both in terms of data supported versus data unsupported variables but also the lossless and lossy compression of variables.

APPENDIX

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

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

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

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

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

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

Proposition 2: The function $w_I(I) = w_I(z_I(I))$, where $z_I = (1-\beta)i_l + \beta i_u$ is a valid weight function in Eq. 8 if it has the following properties: (i) $\beta \in [0, 1]$, (ii) $h : [0, 1] \to [0, 1]$ and (iii) $h(z_I)$ is non-decreasing.

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

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

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

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

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

REFERENCES

[1] M. Sugeno, "Fuzzy measure and fuzzy integral," *Fuzzy Measure and Fuzzy Integral*, vol. 8, no. 2, pp. 94–102, 1972.

[2] G. Choquet, "Theory of capacities," *Annales de l'Institut Fourier*, vol. 5, pp. 131–295, 1954.

[3] P. Melin, O. Mendoza, and O. Castillo, "Face recognition with an improved interval type-2 fuzzy logic sugeno integral and modular neural networks," *IEEE Trans. Syst., Man, Cybern. A*, vol. 41, no. 5, pp. 1001–1012, Sep 2011.

[4] H. Tahani and J. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 733–741, 1990.

[5] S. Cho and J. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 2, pp. 380–384, 1995.

[6] M. Grabisch and J. Nicolas, "Classification by fuzzy integral: performance and tests," *Fuzzy Sets Syst.*, vol. 65, no. 2/3, pp. 255–271, 1994.

[7] M. Grabisch and M. Sugeno, "Multi-attribute classification using fuzzy integral," in *Proc. IEEE Int. Conf. Fuzzy Syst.* IEEE, Mar 1992, pp. 47–54.

[8] M. Grabisch, "Fuzzy integral for classification and feature extraction," *Fuzzy Measures and Integrals: Theory and Applications*, pp. 415–434, 2000.

[9] A. Mendez-Vazquez, P. Gader, J. Keller, and K. Chamberlin, "Minimum classification error training for Choquet integrals with applications to landmine detection," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 225–238, Feb 2008.

[10] R. Yang, Z. Wang, P. A. Heng, and K. S. Leung, "Classification of heterogeneous fuzzy data by Choquet integral with fuzzy-valued integrand," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 5, pp. 931–942, Oct 2007.

[11] J. M. Keller, P. Gader, H. Tahani, J. Chiang, and M. Mohamed, "Advances in fuzzy integration for pattern recognition," *Fuzzy Sets Syst.*, vol. 65, no. 2-3, pp. 273–283, 1994.

[12] G. Beliakov, S. James, and G. Li, "Learning Choquet-Integral-Based metrics for semisupervised clustering," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 3, pp. 562–574, Jun 2011.

[13] P. D. Gader, J. M. Keller, and B. N. Nelson, "Recognition technology for the detection of buried land mines," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 1, pp. 31–43, 2001.

[14] M. Grabisch, "The application of fuzzy integrals in multi-criteria decision-making," *Eur. J. Oper. Res.*, vol. 89, no. 3, pp. 445–456, 1996.

[15] C. Labreuche, "Construction of a Choquet integral and the value functions without any commensurateness assumption in multi-criteria decision making," in *EUSFLAT Conf.*, 2011, pp. 90–97.

[16] J. Chiang, "Choquet fuzzy integral-based hierarchical networks for decision analysis," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 63–71, Feb 1999.

[17] A. F. Tehrani, W. Cheng, and E. Hullermeier, "Preference learning using the Choquet integral: The case of multipartite ranking," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, pp. 1102–1113, Dec 2012.

[18] S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo, "Assessing non-additive utility for multicriteria decision aid," *Eur. J. Oper. Res.*, vol. 158, no. 3, pp. 734–744, Nov 2004.

[19] S. Angilella, S. Greco, and B. Matarazzo, "Non-additive robust ordinal regression: A multiple criteria decision model based on the Choquet integral," *Eur. J. Oper. Res.*, vol. 201, no. 1, pp. 277–288, Feb 2010.

[20] S. Angilella, S. Corrente, and S. Greco, "Stochastic multiobjective acceptability analysis for the Choquet integral preference model and the scale construction problem," *Eur. J. Oper. Res.*, vol. 240, no. 1, pp. 172–182, Jan 2015.

[21] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. F. Anderson, and D. J. Wescott, "Extension of the fuzzy integral for general fuzzy set-valued information," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 6, pp. 1625–1639, Dec 2014.

[22] A. F. Tehrani, W. Cheng, K. Dembczyski, and E. Hllermeier, "Learning monotone nonlinear models using the Choquet integral," *Machine Learning*, vol. 89, no. 1-2, pp. 183–211, Oct 2012.

[23] A. F. Tehrani, W. Cheng, and E. Hüllermeier, "Choquistic regression: Generalizing logistic regression using the Choquet integral." in *EUSFLAT Conf.*, 2011, pp. 868–875.

[24] A. J. Pinar, J. Rice, L. Hu, D. T. Anderson, and T. C. Havens, "Efficient multiple kernel classification using feature and decision level fusion," *IEEE Trans. Fuzzy Syst.*, pp. 1–1, 2016.

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

[26] J. M. Keller and J. Osborn, "Training the fuzzy integral," *International Journal of Approximate Reasoning*, vol. 15, no. 1, pp. 1–24, Jul 1996.

[27] ——, "A reward/punishment scheme to learn fuzzy densities for the fuzzy integral," *Proc. Int. Fuzzy Sys. Assoc. World Cong*, pp. 97–100, 1995.

[28] A. Mendez-Vazquez and P. Gader, "Sparsity promotion models for the Choquet integral," in *2007 IEEE Symp. Found. Comput. Intell.* IEEE, Apr 2007, pp. 454–459.

[29] G. Beliakov, "Construction of aggregation functions from data using linear programming," *Fuzzy Sets Syst.*, vol. 160, no. 1, pp. 65–75, Jan 2009.

[30] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.

[31] M. Grabisch, "K-order additive discrete fuzzy measures and their representation," *Fuzzy Sets Syst.*, vol. 92, no. 2, pp. 167–189, 1997.

[32] H. Bustince, G. Beliakov, G. P. Dimuro, B. Bedregal, and R. Mesiar, "On the definition of penalty functions in data aggregation," *Fuzzy Sets*

Syst., 2016.
[33] D. T. Anderson, S. R. Price, and T. C. Havens, "Regularization-based learning of the Choquet integral," in *2014 IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*. IEEE, Jul 2014, pp. 2519–2526.
[34] P. Benvenuti, R. Mesiar, and D. Vivona, *Monotone Set Functions-Based Integrals*. Elsevier, 2002, pp. 1329–1379.
[35] M. Grabisch, I. Kojadinovic, and P. Meyer, "A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the kappalab r package," *Eur. J. Oper. Res.*, vol. 186, no. 2, pp. 766–785, 2008.
[36] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM J. Numerical Anal.*, vol. 17, pp. 238–246, 1980.

Muhammad Aminul Islam received the B.Sc. degree in electrical and electronic enginnering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2005. He is currently pursuing the Ph.D. degree in electrical and computer engineering (ECE) at Mississippi State University, Mississippi State, MS, USA.

He is presently an Instructor in ECE at Mississippi State University. His research interests include data/information fusion, machine learning, and hyperspectral image processing.

Derek T. Anderson received the Ph.D. in electrical and computer engineering (ECE) in 2010 from the University of Missouri, Columbia, MO, USA.

He is currently the Robert D. Guyton Chair Associate Professor in ECE at Mississippi State University (MSU), USA, an Intermittent Faculty Member with the Naval Research Laboratory and co-director of the Sensor Analysis and Intelligence Laboratory (SAIL) at MSU. His research interests include data/information fusion for machine learning and automated decision making in signal/image understanding and computer vision.

Dr. Anderson co-authored the 2013 best student paper in Automatic Target Recognition at SPIE, received the best overall paper award at the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) 2012, and he received the 2008 FUZZ-IEEE best student paper award. Dr. Anderson is the Program Co-Chair of FUZZ-IEEE 2019 and an Associate Editor for the IEEE Transactions on Fuzzy Systems.

Anthony J. Pinar received the Ph.D. degree in electrical and computer engineering in 2017 from Michigan Technological University, Houghton, MI, USA, where he is currently a lecturer.

His research interests include data/information fusion, machine learning, autonomous robotics, and signal processing.

Timothy C. Havens received the Ph.D. degree in electrical and computer engineering from the University of Missouri, Columbia, MO, USA, in 2010.

He is currently the William and Gloria Jackson Associate Professor with the Department of Electrical and Computer Engineering and Department of Computer Science, Michigan Technological University, Houghton, MI, USA. He is the Director of the ICC Center for Data Sciences and Executive Director of the MTU Data Science Program.

Dr. Havens was a recipient of the Best Paper Award at the 2012 IEEE International Conference on Fuzzy Systems, the IEEE Franklin V. Taylor Award for Best Paper at the 2011 IEEE Conference on Systems, Man, and Cybernetics, and the Best Student Paper Award from the Western Journal of Nursing Research in 2009. Dr. Havens is the General Chair of 2019 IEEE International Conference on Fuzzy Systems. He is an Associate Editor of the IEEE Transactions on Fuzzy Systems.