Multi-Criteria Based Learning of the Choquet Integral using Goal Programming

Muhammad Aminul Islam and Derek T. Anderson
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762
Email: (mi160,anderson)@msstate.edu

Timothy C. Havens
Department of Electrical and Computer Engineering
and Computer Science
Michigan Technological University
1400 Townsend Dr.
Houghton, MI 49931
Email: thavens@mtu.edu

***Abstract*—In this paper, we explore a new way to learn an aggregation operator for fusion based on a combination of one or more labeled training data sets and information from one or more experts. One problem with learning an aggregation from training data alone is that it often results in solutions that are overly complex and expensive to implement. It also runs the risk of over-fitting and the quality of that solution is based in large on the size and diversity of the data employed. On the other hand, learning an aggregation based on only expert opinions can be overly subjective and may not result in desired performance for some given task. In order to overcome these shortcomings, we explore a new way to combine both of these important sources. However, conflict between data sets, experts or a combination of the two, can (and often do) occur and must be addressed. Herein, weighted Goal programming, an approach from *multi-criteria decision making* (MCDM), is employed to learn the *fuzzy measure* (FM) relative to the *Choquet integral* (CI) for data/information fusion. This framework provides an interesting way in which we can set the priority order of any number of combination of these two sources. Furthermore, it provides a mechanism to preserve the monotonicity constraints of the FM. We demonstrate results from synthetic experiments across a range of different conflicting and combination of sources scenarios.**

I. Introduction

The *fuzzy integral* (FI) has been used time and time again in numerous applications such as signal and image processing, pattern recognition and *multi-criteria decision making* (MCDM) for data/information aggregation. In 1974, Sugeno introduced the *fuzzy measure* (FM), a monotone and normal capacity [1]. The FM is significant with respect to the FI because it is in effect what drives (determines what specific function is computed) nonlinear aggregation via FIs like the *Choquet integral* (CI) and *Sugeno integral* (SI). Existing approaches either manually specify the FM or attempt to learn it from data according to a criteria such as the *sum of squared error* (SSE). However, the FM is difficult to specify as it has $(2^N - 2)$, for N inputs, numbers of "free parameters". It is extremely rare than an expert can (or would want to) provide such information even for relatively small cases, e.g., $N = 4$ has 14 values, $N = 5$ has 30 and $N = 10$ already has $1,022$ values. Common practice is to specify just the *densities*, the measure values for only the singletons (individuals). From there, we can use methods such as the S-Decomposable FM, the Sugeno FM [1], and Grabischs k-additive FM and integral [2]. In [3], Grabisch and Roubens proposed a method known as constraint satisfaction that takes the relative importance of the criteria and the type of interaction between them, if any. Alternately, identification of FMs based on data has been explored by numerous authors in different applications; *quadratic programming* (QP) [4], gradient descent [5], penalty and reward [6], Gibbs sampler [7], and genetic algorithms [8]. However, to the best of our knowledge no method has been put forth to date to learn the FM by taking into account a weighted combination of both experts' knowledge and data.

One problem with learning an aggregation from data alone is that is often results in solutions that are overly complex and "expensive" to implement. It also runs the risk of over-fitting and the quality of that solution is based in large on the size and diversity of the data. On the other hand, learning an aggregation based on only expert opinions can be overly subjective and may not result in peak performance for some given task. However, conflict between data sets, experts or a combination of the two, can (and often do) occur and must be addressed. Consider the following simple but real example of robust multi-sensor based target detection. The goal is to utilize the individual (and most important, combined) benefit of different sensors operating in different parts of the electromagnetic spectrum. While we want to achieve the highest performance possible, we also desire a solution whose overall cost does not exceed some limit, one that requires the fewest computational resources (e.g., memory and processing), has the smallest form factor and energy consumption, etc. At the heart of such a dilemma is the need to optimize some process relative to different and often conflicting information.

Herein, weighted Goal programming [9], an approach from MCDM, is employed to learn the FM relative to the CI for flexible data/information fusion. This mathematical framework provides an interesting and new way in which we can set the priority order of any number of combination of different sources (e.g., data and high-level expert knowledge) in learning. Specifically, relative weights are placed on data and experts. In this way, the user can exercise control on how and to what extent the criteria from each input contribute to learning. Figure 1 shows the proposed weighted Goal programming framework for CI learning.

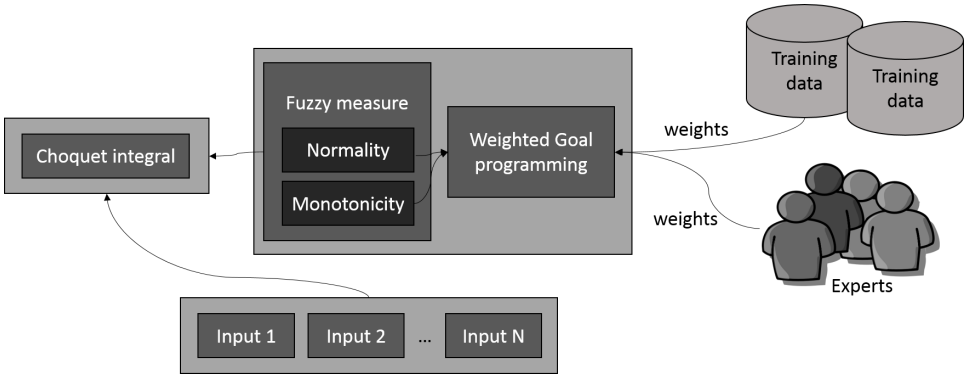

Fig. 1. Proposed weighted Goal programming framework to learn the CI from a combination of data and high-level expert knowledge. Relative priorities (weights) are placed on training data and experts. Weighted Goal programming also allows for a way to enforce the hard constraints imposed by the fuzzy measure (normality and monotonicity).

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

II. Background

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

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

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

Note, often $g(X) = 1$ is enforced as an additional boundary constraint. Also, if X is an infinite set then a third condition guaranteeing continuity is required, but this is a moot point for finite X. For example, for three inputs, we have the following boundary constraints

$$g(\phi) = 0,$$
$$g(X) = 1,$$

and the montonocity constraints are

$$g_1 - g_{12} \leq 0,$$
$$g_1 - g_{13} \leq 0,$$
$$\cdots,$$
$$g_{23} - g(X) \leq 0,$$

where $g_{1,\cdots,i}$ is a lexicographically encoded representation (short hand notation), i.e., $g_1 = g(x_1), g_{13} = g(\{x_1, x_3\})$ etc. Next, we review some notation for training data necessary to define the CI.

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

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

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

Definition 3. (CI) The discrete CI for object O_j is

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

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

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

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

Next, we proceed to the formulation of the SSE minimization problem for FM learning and the QP. Let the SSE between training data T and the CI with respect to the FM g, be [10]

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

which can be expanded as

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

where

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

and

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

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

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

where

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

The CI is learned through minimizing an objective function such as Equation (1) with respect to the monotonicity and boundary conditions of the FM. For N inputs, the FM has 2^N variables, of which two variables $g(\emptyset)$ and $g(X)$ have constant values i.e., $g(\emptyset) = 0$, and $g(X) = 1$. The monotonicity constraints are constructed with the remaining $2^N - 2$ variables without considering these two constant valued variables. The monotonicity constraints for N sources can be represented in matrix form (of size $N(2^{N-1} - 1) \times (2^{N-1} - 1)$) by [10],

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

where,

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

Here, $\boldsymbol{\Psi}_1$ is the vector representing the coefficients of the monotonicity constraint 1, $g(1) - g(12) \leq 0$. So, the coefficient matrix of monotonicity constraints consists of $\{0, 1, -1\}$ [10]

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

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

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

subject to

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

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

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

III. Methods

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

A. Introduction to weighted Goal programming

Goal programming is a technique from MCDM for deriving satisfying solutions from conflicting multi-objective Goals subject to a set of constraints. As for conflicting Goals, simultaneous achievement of all Goals are not possible. Therefore, Goal programming tries to attain each goal as close as possible. This is accomplished by introducing deviation variables in each goal and defining an objective to minimize a function formed with deviation variables or deviation control parameters. Interested readers seeking in-depth knowledge and examples on Goal programming may explore [11]. Consider a simple example of conflicting multiple objectives-profit maximization and cost minimization–of a firm. Assume, the objectives of the problems are:

Profit:

$$\text{maximize } z_1 = 2x_1 + 5x_2 \tag{2a}$$

Expense:

$$\text{minimize } z_2 = 3x_1 + x_2 \tag{2b}$$

Satisfy:

$$2x_1 + x_2 \leq 50 \text{ (resource constraint)} \tag{2c}$$
$$x_1 \leq 25 \text{ (product 1)} \tag{2d}$$
$$x_2 \leq 15 \text{ (product 2)} \tag{2e}$$
$$x \geq 0 \tag{2f}$$

In this example, equations (2c)-(2f) are hard constraints. As both objectives are conflicting, there is no way to find out a solution that satisfies both objectives. Therefore, we set Goal for each of the objectives in an aim to attain the targets as close as possible. Assume, target for profit is 40 and expenses 80. The solution will achieve a profit short of, and expense over the target. Let the shortage amount be denoted by "slack variable" p1 and the excess by n1, and both objectives have equal priority. Now, the Goal programming problem can be defined as:

$$\text{minimize } z = n1 + p2$$

subject to:

$$3x_1 + x_2 + n1 \geq 90$$

$$2x_1 + 3x_2 - p2 \leq 112$$

$$x_1 \leq 20$$

$$x_2 \leq 30$$

$$x1 \geq 0, \; x2 \geq 0.$$

In the above problem, the respective solution is, $x_1 = 20$, $x_2 = 24$, profit is equal to 84 and expense is equal to 130.

Goal programming has several variants, such as Lexicographic Goal programming, Weighted Goal programming and Chebysheve Goal programming [11]. In Lexicographic Goal programming, the goals, categorized into priority order, are optimized in such a way that the goals achieved for the higher priority level must not be worsened while optimizing the next lower priority Goals. In Weighted Goal programming, the decision maker expresses the preference on goals by relative weights and the undesired deviation of each goal are minimized according to these weights. Additional details of these techniques can be found at [11]. Weighted Goal programming has been chosen over other variants because it gives the flexibility to set priority quantitatively to any proportion of choice.

B. Minimization of the SSE

As discussed in the previous section, the CI with respect to training data can be learned by minimizing a objective function that corresponds to the SSE between the CI and the label of the data. Using the SSE function defined in equation (1), the minimization problem can be written as

$$\min_{\mathbf{u}} \sum_{j=1}^{m} \left(\mathbf{A}_{O_j}^t \mathbf{u} - \alpha_j \right)^2 \tag{3}$$

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

As the Goal programming works on constraints with targets to be attained the proximity as nearest as possible, we have to transform the SSE minimization objective in (3) to a set of Goal constraints. The conditions for the minimum value of this convex function is obtained by computing the gradient of this function along all the hyper-axes and then equating the gradients to zero. We start by expanding Equation (3),

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

and we take its derivative,

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

The equality Goal constraints are therefore

$$\sum_{j=1}^{m} \mathbf{A}_{O_j} \mathbf{A}_{O_j}^t \mathbf{u} = \sum_{j=1}^{m} \mathbf{A}_{O_j} \alpha_j, \tag{4}$$

or

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

where

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

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

C. Expert Opinion

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

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

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

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

$$[1 \quad -1 \quad \cdots \quad 0]^t.$$

All the inequality constraints result in the matrix

$$\mathbf{C}_E\mathbf{u} \geq \mathbf{0},$$

where

$$\mathbf{C_E} = \begin{pmatrix} \mathbf{\Phi}_1^t \\ \mathbf{\Phi}_2^t \\ \cdots \\ \mathbf{\Phi}_P^t \end{pmatrix},$$

for P inequality constraints.

Typically, it is far easier, and more realistic, for an expert to qualitatively express knowledge between different subsets of sources in terms of inequalities versus assigning specific values to the different subsets (aka equality constraints). Nevertheless, our mathematical formulation can accommodate for equality constraints if present. If an expert specifies that the "worth" of source 3 is 0.1, then we can simply express this as $g(x_3) = 0.1$. This knowledge can be represented in vector form as

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

where

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

and

$$b = 0.1.$$

Here, a 1 is put in 3rd position in Υ_i^t which corresponds to g_3 in $\mathbf{u}$. If we have equality inputs, then we can pack this into matrix form as

$$\mathbf{C}_{eq}\mathbf{u} = \mathbf{b}$$

where

$$\mathbf{C_{eq}} = \begin{pmatrix} \Upsilon_1^t \\ \Upsilon_2^t \\ \cdots \\ \Upsilon_Q^t \end{pmatrix},$$

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

D. Goal programming for training data and multiple experts

In this subsection, the Goal programming problem is formulated for the case of one or more training data sets and multiple expert's opinion with a priority order represented by relative weights. In standard weighted Goal programming, the objective function consists of single goal minimization of the weighted sum of the undesired deviation. In [12], the author proposed an improved weighted Goal programming method that takes relative normalized weight for the priority of the criteria as well as maintains the relative priority of each goal. The objective function now consists of two goals. The first goal, i.e., term corresponds to the maximum unwanted deviation, and the second term corresponds to sum of the undesired deviations. This method involves lexicographical minimization of the two terms, i.e., minimization of the first term with highest priority and then minimization of the second term while not worsening the first goal. Let the normalized weight on the training data be w_D and the weights on multiple (P) experts be $w_E = \{w_{E1}, w_{E2}, \cdots, w_{EP}\}$ with the normality constraint $w_D + \sum_{l=1}^{P} w_{El} = 1$. The weighted Goal program [12] is as follows,

$$\text{Lex}\min z = \left\{ \beta, \sum_i (n_i + p_i) + \sum_l \left(\sum_j n_{ij} + \sum_k (n_{lk} + p_{lk}) \right) \right\}, \tag{5a}$$

subject to

$$\mathbf{c}_{D_i}\mathbf{u} + n_i - p_i = b_{D_i}, i = 1, 2, ..., 2^N - 2, \tag{5b}$$

$$\mathbf{c}_{El_j}\mathbf{u} + n_{lj} \geq b_{Ep_{lj}}, l = 1, 2, ..., P, \tag{5c}$$

$$\mathbf{c}_{Eeql_k}\mathbf{u} + n_{lk} - p_{lk} = b_{Eeql_k}, l = 1, 2, ..., P, \tag{5d}$$

$$\mathbf{C}_M\mathbf{u} \geq \mathbf{0}, \tag{5e}$$

$$w_D(n_i + p_i) \leq \beta, \tag{5f}$$

$$w_{El}n_{lj} \leq \beta, \tag{5g}$$

$$w_{El}(n_{lk} + p_{lk}) \leq \beta, \tag{5h}$$

$$n_i, p_i, n_{ij}, n_{ik}, p_{ik} \geq 0, \tag{5i}$$

$$(\mathbf{0}, 1)^t \leq \mathbf{u} \leq \mathbf{1}, \tag{5j}$$

where $\mathbf{c}_{D_i}$ and b_{D_i} corresponds to the i-th row of the coefficient matrix and target vector of the training data constraints. Similar definition applies to $\mathbf{c}_{El_m}$, which is the m-th row of the coefficient matrix of l-th expert's inequality constraints. The parameter β minimizes the maximum undesired deviation while maintaining the priority order of the constraints. Equations (5e) to (5j) are the hard constraints which are always satisfied. The method in [11] applies normalization on each criteria or constraints with respect to target. However, we have not normalize the target values of criteria (e.g., b_{D_i}) since the target values are either zero or close to zero.

In the above we restricted the mathematical formulation for learning the CI to a single data set and multiple experts. However, we can utilize the underlying concept of the weighted Goal programming to generalize the framework to cover any number of data sets and multiple experts. For this more general case, there are (at least) two strategies for assigning relative weights. In the first approach, the priority order or relative weight assignment can be similar to that of a single data set and multiple experts. For example, if we have two data sets and two experts, then the relative weights could be 0.2, 0.3, 0.1, and 0.4 on data set 1, data set 2, expert 1 and expert 2 respectively. So, in terms of weights, it does not differentiate between data sets and experts.

In the second approach we can prioritize the group of data sets and group of experts first. At the next level, each member

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

Next, we generate the equality constraints for all the training data sets as given by (4), equality and equality constraints for each expert according to (**??**) and (**??**). With all the constraints in hand, it is simple to formulate the weighted Goal programming problem in which the objective is to achieve the goal of minimizing the deviation from the goal of each criteria according to their relative priorities. Depending on the weight assignment strategy, the objective function will optimize for all data sets and experts in a single level weight case or we sequentially minimize the deviation of goal for groups (e.g., group of data sets) then for each criteria (e.g., each data set).

IV. Experiments and Results

In this section, we illustrate the proposed method via different synthetic experiments. Synthetic experiments are used in order to illustrate the behavior of the technique under different extreme and informative scenarios. First, we set the context by discussing different possible applications for the proposed work. Then, we show results for six different cases and we discuss their results.

Example 1. Consider the following scenario from multi-sensor signal/image processing. Let there be three sensors available to detect some object/target and assume there are three algorithms, one for each sensor. Without loss of generality, this example is easily extended to the case of a single (or multiple) sensor(s) with different algorithms or the same algorithm under different parameters or trained differently. Assume we have both training data and multiple expert information available. Furthermore, the user has placed a (relative) priority (via weighting) of 0.8 on the training data, in order to learn a solution that will primary minimize function error and hopefully perform well on a given domain (if the training data/solution generalized well to new data obviously), while expert 1 (E1) is given a weighting (w_{E1}) of 0.1 and expert 2 (E2) is given a weighting (w_{E2}) of 0.1 (thus equal expert worth). E1 has information regarding the complexity, e.g., computationally and/or memory utilization wise, of the respective algorithms. This information is important because it impacts factors like processing hardware (which impacts the financial cost of the system) and ultimately it determines the system's response time. E1 can provide this information in the form of partial order relations about the relative worth of the algorithms; e.g., $g_4 \succeq g_1$,and $g_{12} \succeq g_3$. The constraints constructed from E1's criteria is

$$g_4 - g_1 \geq 0,$$
$$g_{12} - g_3 \geq 0,$$

and the size of the coefficient matrix is $\mathbf{C}_{E1}$ is $2x(2^4-2)$. On the other hand, E2 has knowledge regarding the financial cost of the different sensors and he/she expresses this knowledge in terms of $E2 = \{g_4, g_2, g_1, g_3\}$. The constraints from E2 are therefore,

$$g_4 - g_2 \geq 0,$$
$$g_2 - g_1 \geq 0,$$
$$g_1 - g_3 \geq 0.$$

This information is important obviously because it ultimately impacts our ability to produce and use such a technology. Note, we have expressed E1 and E2's information as partial order inequalities, however if exact numbers are know or can be produced and if a user wishes to convert them into specific FM values then this can be done in our framework via equality constraints. From the above it is apparent, all inputs are important and no single input is sufficient to solve this task. We would like to have a solution that has the fewest number of inputs and least processing time that is optimal for a given problem/domain. It is also apparent that conflicts could easily arise between what is the fastest algorithm, the lowest cost sensor and what is best for the training data.

Example 2. In example 2, consider the scenario of two training data sets and no expert information. We use the field of computer vision and hyperspectral image processing to motivate this example. Hyperspectral image processing is common in scenarios like remote sensing. Ideally, hyperspectral sensors operate by gathering large amounts of data about their field of view (FOV) by sub-dividing the part of the electromagnetic spectrum that is often referred to as light into a relatively large number of evenly-spaced, narrow-wavelength bands. Often, there are typically hundreds or thousands of bands collected by a hyperspectral sensor, which typically cover ultraviolet, visible, near-infrared, and sometimes long-wave infrared (thermal). When a hyperspectral image is created, spectral band measurements are collected in the spatial region covered by each pixel. Thus, an extremely large amount of information is collected in a single hyperspectral image. One of the things that makes hyperspectral imaging particularly exciting, relative to past imaging sensors, is the ability to deeply analyze the results even at the pixel level. In terms of computer vision, and ultimately hyperspectral image processing for this example, one typically utilizes machine learning algorithms to train one or more classifiers and fusion is used to combine the evidence from these different classifiers (on all of the bands or subsets of the bands) to achieve more robust solutions. Examples include low level feature level fusion, e.g., multiple kernel theory, and high level decision level fusion, e.g., ensemble theory or *multiple classifier decision fusion* (MCDF) such as majority votes. Often, when training an algorithm we include as much training data as available. However, every collection is not created equal. Depending on numerous factors, such as solar radiation and atmospheric conditions, some data collections are better than others. In a field such as precision agriculture, there are also factors such as the time of year and weather

that we must consider. Our goal is to include as much data as possible for training, however if the "quality" of the data varies then we can assign different priority (via weightings). Additionally, in hyperspectral image processing it is also not trial to label data, i.e., assign class labels. In the case of aerial systems, each pixel has a *ground sampled distance* (GSD) and most systems have spectral mixing, which means that each pixel in reality is a mixture of different physical materials. This makes labeling hyperspectral imagery difficult, namely in regions around the boundary of class labels, but neverless througout the data. Information such as this could be taken into account via our procedure and different sets of varying level quality labels could be assigned and used in training a solution.

The above examples are just two of many possible scenarios in which the proposed technique could be applied. In the remainder of this section, we explore different extreme conditions instead of just a single application. This is done in order to keep the problem tractable and the benefit is if we know the answer and all of them information then we can better study how the system reacts instead of analyze what is otherwise a black box. We generated a training data set by generating (in Matlab) m pseud-randomly generated training instances with $N = 3$ inputs each. We use an *ordered weighted average* (OWA) with weights $\{0.7, 0.3, 0\}$ to generate our labels. Thus, we have a FM with values; $g_1 = .7$, $g_2 = .7$, $g_3 = .7$, $g_{12} = 1$, $g_{13} = 1$, $g_{23} = 1$. For a (weighting) vector $w = (w_1, ..., w_N)$, where $w_i \in [0, 1]$ and $\sum_{i=1}^{N} w_i = 1$, consider a permutation (π) on our inputs such that $h\left(O_j : x_{\pi(1)}\right) \geq h\left(O_j : x_{\pi(2)}\right) \geq ... \geq h\left(O_j : x_{\pi(N)}\right)$. An OWA is calculated according to

$$w_1 h\left(O_j : x_{\pi(1)}\right) + ... + w_N h\left(O_j : x_{\pi(N)}\right).$$

The following scenarios involve different combinations of multiple expert inputs with varying weights and conflict. Table I reports the results for each of the following cases, which are outlined and discussed in detail in the following subsections.

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

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

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

4) Multiple experts: In this scenario, rows 5 and 6 from Table I, we use multiple experts and training data. Specifically, let the criteria from the first expert be $g_1 \geq g_{23}$ and the second expert be $g_1 \geq g_{23}$. Thus, both experts have conflicting criteria between themselves as well as with the training data. In both scenarios, we let the importance of the training data set be relatively low and we vary the importances of the expert information. In the case that we place more weight on E1, we see that g_1 is now higher than g_{23}. In the case that E2 is more important, then we can see that $g_3 \geq g_{12}$. Again, each solution (FM) is a complex combination of different weighted accounts of expert information and training data. However, it is important to note that in each of the above cases our soft constraints were taken into account while our hard constraint, the boundary and monotonicity constraints, are enforces (ortherwise we would not have learned a FM).

V. Conclusion

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

In future work, we will expand the work and consider how to include goals of FM complexity. For example, we showed that regularization is a useful way to learn low compleixty FMs. We will include this information into the learning process and we will develop an open source library for researchers. Furthermore, we will apply the concepts put forth herein on different real world problems, e.g., multi-sensor fusion.

References

[1] M. Sugeno, "Theory of fuzzy integrals and its applications," Ph.D. dissertation, Tokyo Institute of Technology, Tokyo, Japan, 1974.

TABLE I
LEARNED FM FOR DIFFERENT SCENARIOS INVOLVING TRAINING DATA, MULTIPLE EXPERTS AND CONFLICT.

Scenario	weights	g_1	g_2	g_3	g_{12}	g_{13}	g_{23}
Training data only	training data = 1, expert = 0	0.7	0.7	0.7	1	1	1
Data and expert with no conflict	training data = 0.3, expert = 0.7	0.7	0.7	0.7	1	1	1
Priority on expert 1 (E1) and conflict	training data = 0.01, expert = 0.99	0.8709	0.8710	0.7335	0.8710	0.9601	0.8710
Priority on training data and conflict	training data = 0.99, expert = 0.01	0.7696	0.9027	0.7204	0.9027	0.9820	0.9027
Multiple experts (priority on E1)	training data = 0.1, E1 = 0.8, E2 = 0.1	0.9198	0.7710	0.7694	0.9198	0.9198	0.7710
Multiple experts (priority on E2)	training data = 0.1, E1 = 0.1, E2 = 0.8	0.7741	0.7741	0.7619	0.7741	0.9395	0.7863

[2] P. Miranda, M. Grabisch, and P. Gil, "Axiomatic structure of¡ i¿, k¡/i¿-additive capacities," *Mathematical Social Sciences*, vol. 49, no. 2, pp. 153–178, 2005.

[3] M. Grabisch and M. Roubens, "Application of the choquet integral in multicriteria decision making."

[4] M. Grabisch, H. Nguyen, and E. Walker, "Fundamentals of uncertainty calculi with applications to fuzzy inference, 1995."

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

[6] ——, "A reward/punishment scheme to learn fuzzy densities for the fuzzy integral," *International Fuzzy Systems Association World Congress*, no. 1, pp. 97–100, 1995.

[7] A. Mendez-Vazquez and P. Gader, "Sparsity promotion models for the choquet integral," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. IEEE, 2007, pp. 454–459.

[8] D. T. Anderson, J. M. Keller, and T. C. Havens, "Learning fuzzy-valued fuzzy measures for the fuzzy-valued sugeno fuzzy integral," in *Computational Intelligence for Knowledge-Based Systems Design*. Springer, 2010, pp. 502–511.

[9] J. P. Ignizio, *Goal programming and extensions*. Lexington Books Lexington, MA, 1976, vol. 26.

[10] D. T. Anderson, S. R. Price, and T. C. Havens, "Regularization-based learning of the choquet integral."

[11] J. P. Ignizio and C. Romero, "Goal programming," *Encyclopedia of information systems*, vol. 2, pp. 489–500, 2003.

[12] M. G. Iskander, "A suggested approach for solving weighted goal programming problem," *American Journal of Computational and Applied Mathematics*, 2012.