Feature and Decision Level Fusion Using Multiple Kernel Learning and Fuzzy Integrals

Anthony Pinar and Timothy C. Havens
Department of Electrical and Computer Engineering
Department of Computer Science
Michigan Technological University
Houghton, MI 49931, USA
e-mail: {ajpinar, thavens}@mtu.edu

Derek T. Anderson and Lequn Hu
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39759, USA
e-mail: anderson@ece.msstate.edu, lh432@msstate.edu

***Abstract*—Kernel methods for classification is a well-studied area in which data are implicitly mapped from a lower-dimensional space to a higher-dimensional space to improve classification accuracy. However, for most kernel methods, one must still choose a kernel to use for the problem. Since there is, in general, no way of knowing which kernel is the best, *multiple kernel learning* (MKL) is a technique used to learn the aggregation of a set of valid kernels into a single (ideally) superior kernel. The aggregation can be done using weighted sums of the pre-computed kernels, but determining the summation weights is not a trivial task. A popular and successful approach to this problem is *MKL-group lasso* (MKLGL), where the weights and classification surface are simultaneously solved by iteratively optimizing a min-max optimization until convergence. In this work, we propose an ℓ_p-normed *genetic algorithm MKL* (GAMKL$_p$), which uses a genetic algorithm to learn the weights of a set of pre-computed kernel matrices for use with MKL classification. We prove that this approach is equivalent to a previously proposed fuzzy integral aggregation of multiple kernels called *fuzzy integral: genetic algorithm* (FIGA). A second algorithm, which we call *decision-level fuzzy integral MKL* (DeFIMKL), is also proposed, where a fuzzy measure with respect to the fuzzy Choquet integral is learned via quadratic programming, and the decision value—viz., the class label—is computed using the fuzzy Choquet integral aggregation. Experiments on several benchmark data sets show that our proposed algorithms can outperform MKLGL when applied to *support vector machine* (SVM)-based classification.**

***Keywords*—multiple kernel learning, Choquet fuzzy integral, fuzzy measure, quadratic programming, genetic algorithm, support vector machine**

TABLE I: Acronyms and Select Notation

SVM	support vector machine
MKL	multiple kernel learning
FM	fuzzy measure
FI	fuzzy integral
FIGA	fuzzy integral: genetic algorithm
LCS	linear convex sum
GAMKL$_p$	p-norm genetic algorithm MKL
DeFIMKL	decision-level fuzzy integral MKL
QP	quadratic program
MKLGL	MKL group lasso
MKLGL$_p$	MKLGL with p-norm regularization
RBF	radial basis function
X	feature-vector data, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{R}^d$
$\mathbf{y}$	data labels, $\mathbf{y} = (y_1, \ldots, y_n)^T$
$f(\mathbf{x})$	prediction function
g	fuzzy measure
$\pi(i)$	sorting index in Choquet integral
$\phi(\mathbf{x})$	non-linear mapping of $\mathbf{x}$
$\kappa(\mathbf{x}_i, \mathbf{x}_j)$	kernel function, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
K	kernel matrix $K = [K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)]$
$f_k(\mathbf{x})$	decision function using kth kernel, K_k
$f_g(\mathbf{x})$	decision function using Choquet integral, wrt FM g

I. Introduction

Consider a set of numerical *feature-vector* data that has the form $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{R}^d$, where the coordinates of $\mathbf{x}_i$ provide feature values (e.g., bits per second, speed, volts, etc.) describing some object (e.g., a wireless sensor network node, traffic camera, or radar). We are also given a set of training labels for each feature vector, such that we have the pair $(\mathbf{y}, X)$, where $\mathbf{y} = (y_1, \ldots, y_n)^T$ and y_i is the label of ith object. Each y_i is associated with a respective feature vector $\mathbf{x}_i$. The classifier learning task is thus to learn some prediction function f, such that we can predict the label of feature-vectors, i.e., $y = f(\mathbf{x})$.

Most classifiers delineate the classes by finding some "best" decision boundary in the feature space. Neural networks and linear *support vector machines* (SVMs) find hyperplanes. These classifiers are easy to train, often can be effective, and are computationally very efficient (the operational decision is just a single dot-product in the feature space). However, they are ineffective for classes that are not linearly separable, i.e., by a hyperplane. Hence, we will use kernel classifiers to non-linearly project the features into a high-dimensional space, where hyperplanes may be more easily found that serve as good decision boundaries.

Specifically, we will focus on *multiple kernel learning* (MKL) in this paper. As its name implies, MKL combines multiple kernel together to form a new kernel, and thus a new space. There are many works that discuss MKL [1–5], and nearly all of them rely on operations that aggregate kernels in ways that preserve symmetry and positive semi-definiteness, such as element-wise addition and multiplication. Most MKL algorithms learn a "best" kernel space in which to classify by learning respective weights on each component kernel. Details are contained in Section III.

The MKL formulations explored in this paper will focus on aggregation using the Choquet *fuzzy integral* (FI) with respect

to a *fuzzy measure* (FM) [6]. First, we will investigate our previously proposed *fuzzy integral: genetic algorithm* (FIGA) approach to MKL [4, 5], proving that it reduces to a special kind of *linear convex sum* (LCS) kernel aggregation. This will lead to the proposition of the *p-norm genetic algorithm* MKL (GAMKL$_p$) approach, which will learn an MKL classifier using a genetic algorithm and generalized p-norm weight domain. These algorithms use a feature-level aggregation of the kernel matrices, producing a new feature representation. We will also propose a decision-level MKL called DeFIMKL, which learns a FM with respect to the Choquet FI to fuse decisions from individual kernel classifiers. The FM is learned from training data with a regularized *quadratic program* (QP) approach [7].

The FI-based MKL approaches will be compared with a leading machine learning MKL method, called MKL *group lasso* (MKLGL) [2] on several benchmark data sets. We will also investigate the behavior of the regularization on the results of DeFIMKL. Section II introduces FMs and FIs, specifically the fuzzy Choquet integral. Section III details the MKL methods. Experimental results are presented in Section IV and we provide concluding remarks and ideas for future work in Section V. Table I contains acronyms and selected notation used in this paper.

II. Fuzzy Measures and Fuzzy Integrals

FIs and FMs have been proposed for many applications and for many types of data, from simple numeric data to intervals and type-2 fuzzy sets [8–18]. While manual specification of the FM works for small sets of sources (there are already 16 possible combinations of sources in the power set of 4 sources), manually specifying the values of the FM for large collections of sources is all but impossible. Thus, automatic methods have been proposed, such as the Sugeno λ-measure [19] and the S-decomposable measure [20], which build the measure from the densities (the worth of individual sources), and genetic algorithm [4, 5, 10, 21], Gibbs sampling [22] and other learning methods [7, 23, 24], which build the measure by using training data. Other works [25–27] have proposed learning FMs that reflect trends in the data and have been specifically applied to crowd-sourcing, where the worth of individuals is not known, but extracted from the data.

A. Fuzzy measures

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

P1. $X \in \Omega$;
P2. For $A \subseteq X$, if $A \in \Omega$, then $A^c \in \Omega$;
P3. If $A_i \in \Omega$, then $\bigcup_{i=1}^{\infty} A_i \in \Omega$.

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

P4. (Boundary conditions) $g(\emptyset) = 0$ and $g(X) = 1$;
P5. (Monotonicity) If $A, B \in \Omega$ and $A \subseteq B$, $g(A) \leq g(B)$.

If Ω is an infinite set, then there is also a third property guaranteeing continuity; in practice and in this paper, Ω is finite and thus this property is unnecessary. The FM values of the singletons, $g(\{x_i\}) = g^i$ are commonly called the *densities*. Figure 1 illustrates the lattice of a FM for the case of $n = 3$.

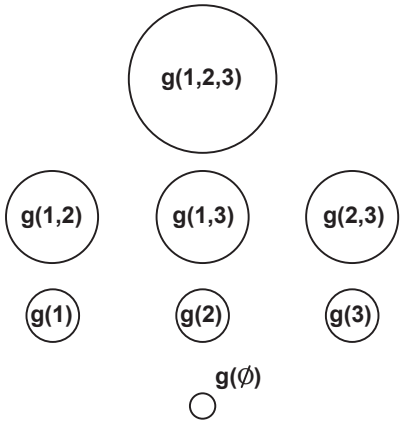

Fig. 1: Lattice of FM elements for $n = 3$. Monotonicity (P5) is illustrated by the size of each circle, i.e., $g(\{x_1\}) \leq g(\{x_1, x_2\})$ as $\{x_1\} \subset \{x_1, x_2\}$.

The arguably most popular FM is the Sugeno λ-measure, which has the attractive property of being able to be defined completely by the values of the densities. The λ-measure has the following additional property. For $A, B \in \Omega$ and $A \cap B = \emptyset$,

$$g_\lambda(A \cup B) = g_\lambda(A) + g_\lambda(B) + \lambda g_\lambda(A) g_\lambda(B), \tag{1a}$$

where it can be shown that λ can be found by solving [19]

$$\lambda + 1 = \prod_{i=1}^{n} \left(1 + \lambda g^i\right), \quad \lambda > 1. \tag{1b}$$

B. Fuzzy integrals

There are many forms of the FI; see [11] for detailed discussion. In practice, FIs are mostly used for evidence fusion [21, 28–31]. They combine sources of information by accounting for both the support of the question (the evidence) and the expected worth of each subset of sources (as supplied by the FM g). Here, we focus on the fuzzy Choquet integral, proposed by Murofushi and Sugeno [32, 33]. Let $h : X \to \mathcal{R}$ be a real-valued function that represents the evidence or support of a particular hypothesis.[1] The discrete (finite Ω) fuzzy Choquet integral is defined as

$$\int_C h \circ g = C_g(h) = \sum_{i=1}^{n} h(x_{\pi(i)}) \left[g(A_i) - g(A_{i-1})\right], \tag{2}$$

where π is a permutation of X, such that $h(x_{\pi(1)}) \geq h(x_{\pi(2)}) \geq \ldots \geq h(x_{\pi(n)})$, $A_i = \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}$, and $g(A_0) = 0$ [6, 14]. Detailed treatments of the properties of FIs can be found in [6, 14, 34]. We now move on to showing how MKL can be achieve using the FM and FI.

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

III. Multiple Kernel Learning

Consider some non-linear mapping function $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x}_i) \in \mathcal{R}^{D_K}$, where D_K is the dimensionality of the transformed feature vector $\phi(\mathbf{x}_i)$. For brevity, we will denote $\phi(\mathbf{x}_i)$ as ϕ_i. With kernel algorithms, one does not need to explicitly transform $\mathbf{x}$, one simply needs to represent the dot product $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The kernel function κ can take many forms, with the polynomial $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p$ and *radial-basis-function* (RBF) $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(\sigma ||\mathbf{x}_i - \mathbf{x}_j||^2)$ being two of the most well known. Given a set of n object vectors X, one can thus construct an $n \times n$ kernel matrix $K = [K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)]_{n \times n}$. This kernel matrix K represents all pairwise dot products of the feature vectors in the transformed high-dimensional space $\mathcal{H}_K$—called the *Reproducing Kernel Hilbert Space* (RKHS).

There are many algorithms that use kernels to transform the data to a space that is appropriate for the goal of the algorithm; in this paper, we focus on kernel-based classification, such as the SVM [35, 36]. Multiple kernel algorithms, such as MKLGL [2] and FIGA [4], take single kernel algorithms a step further by representing the feature-vector with multiple kernels and then combining them to produce a single decision output. The kernel combination can be computed in many ways, as long as the combination results in a Mercer kernel [37]. For the feature-level fusion algorithms in this paper, we will assume that the kernel $\mathcal{K}$ is composed a weighted combination of precomputed kernel matrices, i.e.,

$$\mathcal{K} = \sum_{k=1}^{m} \sigma_k K_k, \tag{3}$$

where there are m kernels and σ_k is the weight applied to the kth kernel. The domain of σ is very important and many MKL implementations only work for a single domain. For example, $\Delta_2 = \{\sigma \in \mathcal{R}^m : \|\sigma\|_2 = 1, \sigma_k \geq 0\}$ is the ℓ_2-norm MKL [1, 3]. MKLGL [2] uses a generalized MKL instantiation that allows for an ℓ_p-norm domain $\Delta_p = \{\sigma \in \mathcal{R}^m : \|\sigma\|_p = 1, \sigma_k \geq 0\}$, simultaneously learning σ and the parameters of an SVM on the resultant kernel $\mathcal{K}$. FIGA [4] generalizes (3) by representing the computation of $\mathcal{K}$ by the Choquet FI,

$$\mathcal{K} = \sum_{k=1}^{m} \left[g(A_k) - g(A_{k-1}) \right] K_{\pi(k)}, \tag{4}$$

where $A_k = \{K_{\pi(1)}, \ldots, K_{\pi(k)}\}$ is a set of kernel matrices sorted by a base-learner quality measure and the FM g is learned by a GA. Next, we show that the FIGA algorithm is actually learning an LCS MKL and is equivalent to (3) with $\sigma \in \Delta_1$; we will then use this new discovery to propose the GAMKL$_p$ algorithm.

A. The GAMKL$_p$ algorithm

The FIGA algorithm produces an MKL classifier by learning a classifier on the composite kernel $\mathcal{K}$ with the Choquet FI as shown in (4). The final classification function is learned on the kernel $\mathcal{K}$, and, in past works, we have used the SVM for this final learner. The basic steps of FIGA are as follows:

1) Compute kernel matrices $K_k = [\kappa_k(\mathbf{x}_i, \mathbf{x}_j)]^{n \times n}$, $k = 1, \ldots, m$;
2) Train a base-learner (e.g., SVM) on each kernel K_k and record the classification accuracy η_k, $k = 1, \ldots, m$;
3) Collect sorting indices π, such that $\eta_{\pi(1)} \geq \eta_{\pi(2)} \geq \ldots \geq \eta_{\pi(k)}$;
4) Use a GA to learn the FM g, such that the classification accuracy of a learner (e.g., SVM) on $\mathcal{K}$ at (4) is maximized.

The fitness of each chromosome in step 4) of FIGA is the classification accuracy of the learner on $\mathcal{K}$, while the genes are $(m-1)$ distinct values of the FM.[2] Because FIGA learns the sorting π once, in step 2), the GA only needs to learn $(m-1)$ FM values, $g(\{K_{\pi(1)}\})$, $g(\{K_{\pi(1)}, K_{\pi(2)}\})$, …, $g(\{K_{\pi(1)}, \ldots, K_{\pi(m-1)}\})$; by property P4, $g(A_0) = 0$ and $g(A_m) = 1$. This leads to Proposition 1

Proposition 1. If the sorting order π is static, then the FIGA Choquet integral at (4) can be rewritten as

$$\mathcal{K} = \sum_{k=1}^{m} \sigma_{\pi(k)} K_{\pi(k)}, \tag{5}$$

where $\sigma_{\pi(k)} = g(A_k) - g(A_{k-1})$.

Proof: Because the sorting is static, the sets A_k are also static; hence, the summation weight on $K_{\pi(k)}$ is the subtraction of the FM values of the same sets (no matter their values). Hence, we can attach a single weight $\sigma_{\pi(k)}$ to each $K_{\pi(k)}$. ∎

Remark 1. Proposition 1 shows that the FIGA kernel composition at (4) is independent of the initial sorting by π because the summation at (5) can be performed in any arbitrary order and give the same result. Hence, step 2) of FIGA is unnecessary.

Proposition 2. The domain of $\sigma_{\pi(k)}$ is Δ_1.

Proof: The ℓ_1 norm of σ is

$$\sum_{k=1}^{m} \sigma_{\pi(k)} = \sum_{k=1}^{m} g(A_k) - g(A_{k-1}) = g(A_m) - g(A_0) = 1. \tag{6}$$

Furthermore, due the monotonicity property (P5) of g, $\sigma_{\pi(k)} = g(A_k) - g(A_{k-1}) \geq 0$. ∎

Remark 2. Proposition 2 shows the domain of σ upon which FIGA learns. Taking Propositions 1 and 2 together shows that FIGA is equivalent to using a GA to learn the weights $\sigma \in \Delta_1$ in the kernel combination at (3).

In light of this discovery, we propose a GAMKL$_p$ algorithm that uses a GA to learn the weights $\sigma \in \Delta_p$ of (3). When $p = 1$, we have shown that this is equivalent to FIGA. However, because of our discoveries in Propositions 1 and 2, we can simplify and generalize FIGA to allow for learning σ in the generalized domain Δ_p. The genes of the GAMKL$_p$ algorithm

[2] In [5], an additional gene was added to indicate different types of FMs and a slightly better performance was noted.

are the values of the m weights of (3), i.e., $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. To ensure the GAMKL$_p$ genes lie in the ℓ_p-norm domain Δ_p, all candidate genes $\tilde{\sigma}$ are ℓ_p-norm normalized to form σ as

$$\sigma = \frac{\tilde{\sigma}}{\sqrt[p]{\sum_{i=1}^{m} |\tilde{\sigma}_i|^p}}. \tag{7}$$

The fitness of each chromosome in GAMKL$_p$ is the 5-fold cross-validation classification accuracy of the learning algorithm—in this paper, an SVM—trained on each chromosome's aggregated $\mathcal{K}$.

Remark 3. While Propositions 1 and 2 show that FIGA is equivalent to GAMKL$_1$, the GAMKL$_p$ algorithm has the additional benefit that the genes of each chromosome are not constrained to be monotonically increasing (as in FIGA). Hence, GAMKL$_p$ is algorithmically more simple.

In Section IV, we will further investigate the performance of GAMKL$_p$ for real-world classification tasks and in comparison with other MKL classification methods. Now we turn to proposing a decision-level MKL fusion method using the fuzzy integral.

B. The DeFIMKL algorithm

Let $f_k(\mathbf{x}_i)$ be the decision-value on feature-vector $\mathbf{x}_i$ produced by the kth classifier in an ensemble. The overall decision of the ensemble is computed by the Choquet integral, where the evidence h is the set of decisions by the classifier ensemble and g encodes the relative worth of each classifier in the ensemble. So, mathematically, the ensemble decision $f_g(\mathbf{x}_i)$ on feature-vector $\mathbf{x}_i$ with respect to the FM g is produced by

$$f_g(\mathbf{x}_i) = \sum_{k=1}^{m} f_{\pi(k)}(\mathbf{x}_i) \left[g(A_k) - g(A_{k-1}) \right], \tag{8}$$

where $A_k = \{f_{\pi(1)}(\mathbf{x}_i), \ldots, f_{\pi(k)}(\mathbf{x}_i)\}$, such that $f_{\pi(1)}(\mathbf{x}_i) \geq f_{\pi(2)}(\mathbf{x}_i) \geq \ldots \geq f_{\pi(m)}(\mathbf{x}_i)$. This is a generalized classifier fusion method that has been explored in many previous works [17, 30, 31, 38].

In [7], we proposed a method to learn the FM g from training data with a regularized *sum-of-squared error* (SSE) optimization, which we now briefly describe. Let the SSE be defined as

$$E^2 = \sum_{i=1}^{n} \left(f_g(\mathbf{x}_i) - y_i \right)^2. \tag{9}$$

It can be shown that (8), as a Choquet integral, can be reformulated as

$$f_g(\mathbf{x}_i) = \sum_{k=1}^{m} \left[f_{\pi(k)}(\mathbf{x}_i) - f_{\pi(k+1)}(\mathbf{x}_i) \right] g(A_k), \tag{10}$$

where $f_{\pi(m+1)} = 0$ [6]. The SSE can thus be expanded as

$$E^2 = \sum_{i=1}^{n} \left(H_{\mathbf{x}_i}^T \mathbf{u} - y_i \right)^2, \tag{11a}$$

where $\mathbf{u}$ is the lexicographically ordered FM g, i.e., $\mathbf{u} = (g(1), g(2), \ldots, g(1 \cup 2), g(1 \cup 3), \ldots, g(1 \cup 2 \cup \ldots \cup m))$, and

$$H_{\mathbf{x}_i} = \begin{pmatrix} \vdots \\ f_{\pi(1)}(\mathbf{x}_i) - f_{\pi(2)}(\mathbf{x}_i) \\ \vdots \\ 0 \\ \vdots \\ f_{\pi(m)}(\mathbf{x}_i) - 0 \end{pmatrix}, \tag{11b}$$

where $H_{\mathbf{x}_i}$ is of size $(2^m - 1) \times 1$ and contains all the difference terms $f_{\pi(k)}(\mathbf{x}_i) - f_{\pi(k+1)}(\mathbf{x}_i)$ at the corresponding locations of A_k in $\mathbf{u}$. We can now fold out the squared term in (11a), producing

$$\begin{aligned} E^2 &= \sum_{i=1}^{n} \left(\mathbf{u}^T H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T \mathbf{u} - 2 y_i H_{\mathbf{x}_i}^T \mathbf{u} + y_i^2 \right) \\ &= \mathbf{u}^T D \mathbf{u} + \mathbf{f}^T \mathbf{u} + \sum_{i=1}^{n} y_i^2, \\ D &= \sum_{i=1}^{n} H_{\mathbf{x}_i} H_{\mathbf{x}_i}^T, \quad \mathbf{f} = -\sum_{i=1}^{n} 2 y_i H_{\mathbf{x}_i}. \end{aligned} \tag{12}$$

Note that (12) is a quadratic function; hence, we can add in the constraints on $\mathbf{u}$, such that it represents a FM, producing a constrained QP. We can write the monotonicity constraint on $\mathbf{u}$, according to properties P4 and P5, as $C\mathbf{u} \leq 0$, where

$$C = \begin{pmatrix} \Psi_1^T \\ \Psi_2^T \\ \vdots \\ \Psi_{n+1}^T \\ \vdots \\ \Psi_{m(2^{m-1}-1)}^T \end{pmatrix} \tag{13}$$

and Ψ_1^T is a vector representation of the monotonicity constraint, $g(1) - g(1 \cup 2) \leq 0$. Hence, C is simply a matrix of $\{0, 1, -1\}$ values of size $(m(2^{m-1} - 1)) \times (2^m - 1)$. See [7] for more details about the form of C. Thus, the full QP to learn the FM $\mathbf{u}$ is

$$\min_{\mathbf{u}} 0.5 \mathbf{u}^T \hat{D} \mathbf{u} + \mathbf{f}^T \mathbf{u}, \quad C\mathbf{u} \leq \mathbf{0}, \quad (\mathbf{0}, 1)^T \leq \mathbf{u} \leq \mathbf{1}, \tag{14}$$

where $\hat{D} = 2D$. We will also test the performance of ℓ_2 and ℓ_1 regularization on the optimization at (14), i.e.,

$$\min_{\mathbf{u}} 0.5 \mathbf{u}^T \hat{D} \mathbf{u} + \mathbf{f}^T \mathbf{u} + \lambda \|\mathbf{u}\|_p, \tag{15}$$

where $p = 1$ for ℓ_1 regularization and $p = 2$ for ℓ_2. Again, see [7] for more discussion on this topic. The QPs at (14) and (15) provide a method to learn the FM $\mathbf{u}$ (i.e., g) from training data. We now propose a method for using this learning method for ensemble learning with kernel SVMs.

We propose that each learner $f_k(\mathbf{x}_i)$ is a kernel classifier, each trained on a separate kernel K_k; here, we will use the

SVM. The SVM classifier decision value is

$$\eta_k(\mathbf{x}) = \sum_{i=1}^{n} \alpha_{ik} y_i \kappa_k(\mathbf{x}_i, \mathbf{x}) - b_k, \tag{16}$$

which is essentially the distance of $\mathbf{x}$ from the hyperplane defined by the learned SVM model parameters, α_{ik} and b_k [35, 36]. Typically, the class label is then computed as $\text{sgn}\{\eta_k(\mathbf{x})\}$. One could use $f_k(\mathbf{x}) = \text{sgn}\{\eta_k(\mathbf{x})\}$ as the training input to the FM learning at (12), but this eliminates information about which kernel produces the largest class separation—essentially, the difference between $\eta_k(\mathbf{x})$ for classes labeled $y = +1$ and $y = -1$. Hence, we remap $\eta_k(\mathbf{x})$ onto the interval $[-1, +1]$, creating the inputs for learning by the sigmoid function,

$$f_k(\mathbf{x}) = \frac{\eta_k(\mathbf{x})}{\sqrt{1 + \eta_k^2(\mathbf{x})}}. \tag{17}$$

Thus, the training data for DeFIMKL are $(\{K_k = [\kappa_k(\mathbf{x}_i, \mathbf{x}_j)], \mathbf{f}_k(X)\}, \mathbf{y})$, $k = 1, \ldots, m$, where K_k are the kernel matrices for each kernel function κ_k, $\mathbf{f}_k(X) = (f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_n))^T$ are the remapped SVM decision values, and $\mathbf{y} = (y_1, \ldots, y_n)$ are the ground-truth labels of $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, respectively. The output of the QP learner is the FM g. A new feature vector $\mathbf{x}$—from a test data set—can be classified by the trained algorithm with the following procedure:

1) Compute the SVM decision values $f_k(\mathbf{x})$ by using (16) and (17);
2) Apply the Choquet integral at (8) with respect to the learned FM g;
3) Compute the class label by $\text{sgn}\{f_g(\mathbf{x})\}$.

We now will demonstrate the MKL algorithms discussed here by applying them to several benchmark data sets.

IV. Experiments

Here we present the results of the GAMKL$_p$ and DeFIMKL algorithms after applying them to benchmark data sets using SVM classifiers; we use LIBSVM to implement the classifiers [39]. Their performance is compared to that of the MKLGL algorithm discussed in Section III.

A. Datasets and Algorithm Parameters

The benchmark UCI data sets shown in Table II are used to evaluate the algorithms. Note that in some cases multiple classes are joined together such that the classification decision is binary. Each experiment consists of 100 trials so the results can be statistically analyzed using a two-sample *t*-test. In each trial, a random draw of 80% of the data is used for training and the remaining 20% is sequestered for testing. Ten RBF kernels are used in each algorithm with respective RBF width σ linearly spaced on the interval defined in Table III; the same RBF parameters are used for each algorithm.

The genetic algorithm in GAMKL$_p$ has a population of 31 chromosomes, where each chromosome is the set of ℓ_p-norm normalized kernel weights. The GA runs for 25 generations using roulette wheel selection and elitism, where the fittest individual is kept from each generation. One-point crossover with a rate of 60% and a mutation rate of 5% are used, where mutation is simply a random perturbation of the chromosome. Fitness is simply the result of 5-fold cross validation of the kernel-SVM accuracy using the kernel weights comprising each individual, where cross validation is used to suppress the effects of over training. Two experiments are performed with each ℓ_p-norm GAMKL$_p$: one with the initial population generated randomly and another where the initial population is also seeded with the result of the MKLGL algorithm.

TABLE III: RBF Kernel Parameter Ranges

Data Set					
Sonar	**Dermatology**	**Wine**	**Ionosphere**	**Ecoli**	**Glass**
[-2.2, -0.5]	[-2.3, 0.2]	[-2.5, 2]	[-2.1, 1.2]	[-3, 3]	[-2, 2]

The only parameter in the DeFIMKL algorithm is the regularization coefficient λ. Once λ is defined, the QPs at (14) and (15) are solved via an interior-point solver to obtain the FM. The results for regularization using the DeFIMKL algorithm are generated using 10 nonzero λs as well as the case where $\lambda = 0$, corresponding to no regularization of (14).

B. Results

The classification accuracies of the GAMKL$_p$, DeFIMKL, and MKLGL algorithms are shown in Table IV along with the standard deviations over the 100 trials. The best results for each data set are shown in bold font; a two-sample *t*-test at a 5% significance level is used to determine the statistically best results—hence, more than one algorithm can be considered as best. We see that at least two versions of the GAMKL$_p$ algorithm have superior performance on each data set, even outperforming the well established MKLGL algorithm on the Sonar data set. In the other data sets, the performances of GAMKL$_p$ and MKLGL fall very close to each other and their differences are statistically insignificant.

The DeFIMKL results show that it is not as promising as the GAMKL$_p$ algorithm, and regularization generally dampens performance. However, at least one version of the DeFIMKL algorithm still appears as a superior result for half of the data sets. Figures 2 and 3 show the results of the DeFIMKL algorithm applied to the Sonar and Dermatology data sets, respectively. The error bars indicate plus/minus one standard deviation over the 200 runs. Figure 2 and Table IV show that the ℓ_1-norm normalized kernel weights found using DeFIMKL had the best performance when $\lambda = 2$ for the Sonar data set; however, the performance of DeFIMKL is inversely proportional to the value of λ for the Dermatology data set. Thus, the best result obtained from the regularized DeFIMKL algorithm occurs when the kernel weights are ℓ_2-norm normalized with $\lambda = 0.5$. On a final note, it is worthwhile to mention that even in the cases where DeFIMKL algorithm did not achieve superior results, it was only beat by approximately 3%.

V. Conclusion

This paper proposes a feature-level fusion algorithm—GAMKL$_p$—that we show is equivalent to a previously de-

TABLE II: UCI Benchmark Data Sets

	Data Set					
	Sonar	**Dermatology**	**Wine**	**Ionosphere**	**Ecoli**	**Glass**
No. of Objects	208	366	178	351	336	214
No. of Features	60	33	13	34	7	9
Binary Classes	{1} vs. {2}	{1,2,3} vs. {4,5,6}	{1} vs. {2,3}	{0} vs. {1}	{1,2,3,4} vs. {5,6,7,8}	{1,2,3} vs. {4,5,6}

TABLE IV: Classification Accuracy Results on Benchmark Data Sets*

	Data Set					
Algorithm	Sonar	Derm	Wine	Ionosphere	Ecoli	Glass
MKLGL_1	83.0 (5.81)	**97.3** (1.99)	**99.6** (0.97)	**95.2** (2.36)	**97.1** (1.71)	**94.5** (3.29)
MKLGL_2	84.6 (5.11)	**97.2** (1.60)	**99.6** (1.02)	**95.5** (2.40)	**97.2** (1.80)	**94.0** (3.53)
GAMKL_1	84.0 (6.00)	**97.1** (1.70)	99.4 (1.16)	94.8 (2.59)	**97.1** (1.93)	**94.0** (3.87)
GAMKL_1^*	84.6 (5.67)	**97.3** (1.75)	**99.6** (1.00)	94.8 (2.53)	96.9 (1.86)	93.3 (3.99)
GAMKL_2	**86.0** (5.64)	**97.1** (1.55)	**99.5** (1.10)	**95.1** (2.29)	**97.5** (1.60)	**94.0** (3.24)
GAMKL_2^*	**86.4** (5.62)	**96.8** (1.84)	99.4 (1.16)	**95.7** (2.39)	**97.4** (1.68)	**94.2** (3.49)
DeFIMKL	78.9 (5.66)	93.2 (3.07)	99.4 (1.17)	92.3 (7.13)	**97.3** (1.77)	91.2 (3.78)
DeFIMKL_1	**84.9** (6.03) $\lambda = 2$	84.2 (4.12) $\lambda = 0.5$	**99.5** (1.10) $\lambda = 0.5$	88.8 (3.26) $\lambda = 4$	91.8 (3.00) $\lambda = 3$	78.1 (6.20) $\lambda = 0.5$
DeFIMKL_2	84.4 (6.82) $\lambda = 1$	87.6 (3.80) $\lambda = 0.5$	**99.7** (0.87) $\lambda = 1.5$	90.0 (3.35) $\lambda = 0.5$	91.9 (3.26) $\lambda = 2.5$	83.1 (4.83) $\lambda = 0.5$

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

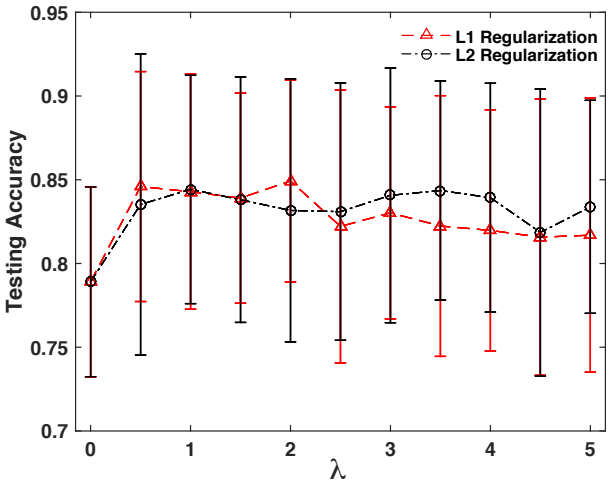

Fig. 2: DeFIMKL performance using regularization on Sonar data. Error bars indicate ± one standard deviation.

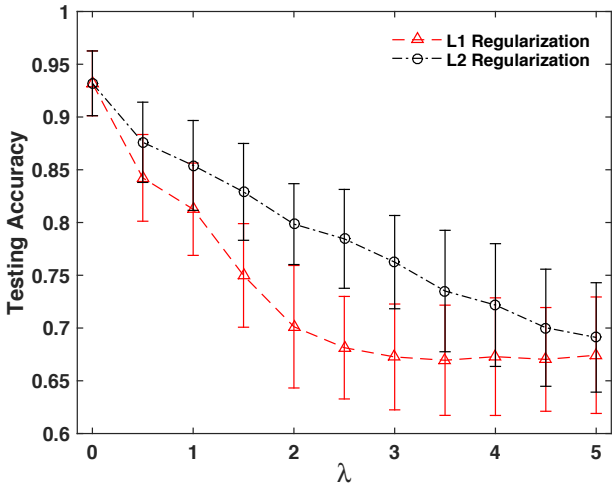

Fig. 3: DeFIMKL performance using regularization on Dermatology data—classes $\{1, 2, 3\}$ versus $\{4, 5, 6\}$. Error bars indicate ± one standard deviation.

veloped fuzzy integral-based MKL approach known as FIGA. However, unlike FIGA, GAMKL_p is generalized such that σ can lie in the ℓ_p-norm domain Δ_p. We also propose a decision-level fusion algorithm—DeFIMKL—that aggregates kernels through the use of the Choquet fuzzy integral with respect to a fuzzy measure learned by a regularized quadratic programming approach. Our results show that the GAMKL_p algorithm achieves equivalent, and sometimes better, classification accuracy than MKLGL. The DeFIMKL algorithm, while generally not as successful as GAMKL_p, still is able to beat MKLGL in half of the experiments, and may be more efficient when applied to distributed systems due to the fact that it combines kernel classifiers at the decision level.

A. *Future work*

Since we discovered that FIGA is equivalent to learning a linear convex sum MKL, we have been working on a feature-level method for aggregating the kernels K_k with a non-linear fuzzy integral. The main goal is to preserve the ability of the fuzzy integral to produce non-linear aggregations of the individual kernels, while ensuring that the result is a Mercer kernel. In order to achieve this, one must develop a way of sorting the kernel matrix terms in the Choquet integral (and not just once with the base-learner training data accuracy, as does FIGA) and still aggregate with a Mercer kernel preserving operation.

Second, the MKL approaches proposed here require the storage of m $n \times n$ kernel matrices, which limits the scal-

ability of these approaches. Furthermore, in the GA-based approaches, a classifier must be trained at each iteration of the GA. We intend to propose an approximation strategy that works to minimize the required storage and the computational complexity of the MKL training. We are experimenting with two solutions. The first involves a linear algebra-based numerical approximation that significantly reduces storage and computational complexity; the second involves developing an integrated solution to simultaneously optimizing the FM/FI aggregation and the resulting MKL classifier.

ACKNOWLEDGEMENTS

This work is funded in part by U.S. Army grants W909MY-13-C-0013 and W909MY-13-C-0029 to support the U.S. Army RDECOM CERDEC NVESD. Dr. Anderson is partially funded by a National Institute of Justice grant (2011-DN-BX-K838), the Army Research Office grants numbered W911NF-14-1-0114 and 57940-EV to support the U.S. Army RDECOM CERDEC NVESD and the Pacific Northwest National Laboratory, under U.S. Department of Energy Contract DE-AC05-76RL01830. Superior, a high performance computing cluster at Michigan Technological University, was used in obtaining some of the results presented in this publication.

REFERENCES

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

[2] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *Proc. Int. Conf. Machine Learning*, 2010, pp. 1175–1182.

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

[4] L. Hu, D. T. Anderson, and T. C. Havens, "Multiple kernel aggregation using fuzzy integrals," in *Proc. IEEE Int. Conf. Fuzzy Systems*, 2013, pp. 1–7.

[5] L. Hu, D. T. Anderson, T. C. Havens, and J. M. Keller, "Validity of different fuzzy integrals and representations for multiple kernel aggregation," in *Proc. Int. Conf. Info. Processing and Management of Uncertainty in Knowledge-Based Systems*, 2014.

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

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

[8] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. F. Anderson, and D. J. Wescott, "Extension of the fuzzy integral for general fuzzy set-valued information," *IEEE Trans. Fuzzy Systems*, 2014.

[9] C. Wagner, D. T. Anderson, and T. C. Havens, "Generalization of the fuzzy integral for discontinuous interval- and non-convex interval fuzzy set-valued inputs," in *Proc. Int. Conf. Fuzzy Systems*, 2013, pp. 1–8.

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

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

[12] D. Zhang and Z. Wang, "Fuzzy integrals of fuzzy valued functions," *Fuzzy Sets and Systems*, vol. 54, pp. 63–67, 1993.

[13] R. Yang, Z. Wang, P. Heng, and K. Leung, "Fuzzified Choquet integral with a fuzzy-valued integrand and its application on temperature prediction," *IEEE Trans. SMC-B*, vol. 38, no. 2, pp. 367–380, Apr. 2008.

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

[15] T. C. Havens, D. T. Anderson, and J. M. Keller, "A fuzzy Choquet integral with an interval type-2 fuzzy number-valued integrand," in *Proc. IEEE Int. Conf. Fuzzy Systems*, Barcelona, Spain, 2010, pp. 1–8.

[16] D. T. Anderson, T. C. Havens, C. Wagner, J. M. Keller, M. Anderson, and D. Wescott, "Sugeno fuzzy integral generalizations for sub-normal fuzzy set-valued inputs," in *Proc. IEEE Int. Conf. Fuzzy Systems*, Brisbane, Australia, 2012, pp. 1–8.

[17] X. Wang, A. Chen, and H. Feng, "Upper integral network with extreme learning mechanism," *Neurocomputing*, vol. 74, no. 16, pp. 2520–2525, 2011.

[18] X. Liang, C. Wei, and Z. Chen, "An intuitionistic fuzzy weighted OWA operator and its applications," *Int. J. Mach. Learn. and Cyber.*, vol. 4, no. 6, pp. 713–719, 2013.

[19] M. Grabisch, T. Murofushi, and M. Sugeno, Eds., *Fuzzy Measures and Integrals. Theory and Applications*. Berlin: Physica Verlag, 2000.

[20] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Application*. New York, NY: Academic Press, 1980.

[21] M. Anderson, D. T. Anderson, and D. Wescott, "Estimation of adult skeletal age-at-death using the Sugeno fuzzy integral," *American Journal of Physical Anthropology*, vol. 142, no. 1, pp. 30–41, 2009.

[22] A. Mendez-Vazquez and P. D. Gader, "Sparsity promotion models for the choquet integral," in *Proc. IEEE Symp. Foundations Comp. Int.*, 2007, pp. 454–459.

[23] J. M. Keller and J. Osborn, "A reward/punishment scheme to learn fuzzy densities for the fuzzy integral," in *Proc. Int. Fuzzy Sys. Assoc. World Cong.*, 1995, pp. 97–100.

[24] ——, "Training the fuzzy integral," *Int. J. Approx. Reasoning*, vol. 15, no. 1, pp. 1–24, 1996.

[25] C. Wagner and D. T. Anderson, "Extracting meta-measures from data for fuzzy aggregation of crowd sourced information," *IEEE International Conference on Fuzzy Systems*, 2012.

[26] T. C. Havens, D. T. Anderson, C. Wagner, H. Deliamsalehy, and D. Wonnacott, "Fuzzy integration of intervals using a measure of generalized accord," in *Proc. IEEE Int. Conf. Fuzzy Systems*, Hyderabad, India, 2013, pp. 1–8.

[27] T. C. Havens, D. T. Anderson, and C. Wagner, "Constructing meta-measures from data-informed fuzzy measures for fuzzy integration of interval inputs and fuzzy number inputs," *IEEE Transactions on Fuzzy Systems*, 2014.

[28] M. Grabisch, *Fuzzy Measures and Integrals: Theory and Applications*. New York: Physica-Verlag, 2000, ch. Fuzzy integral for classification and feature extraction, pp. 415–434.

[29] J. M. Keller, P. Gader, and A. K. Hocaoglu, *Fuzzy Measures and Integrals: Theory and Applications*. New York: Physica-Verlag, 2000, ch. Fuzzy integral in image processing and recognition, pp. 435–466.

[30] S. Auephanwiriyakul, J. M. Keller, and P. Gader, "Generalized Choquet fuzzy integral fusion," *Information Fusion*, vol. 3, pp. 69–85, 2002.

[31] H. Tahani and J. M. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Trans. Systems Man Cybernet.*, vol. 20, no. 3, pp. 733–741, 1990.

[32] G. Choquet, "Theory of capacities," *Analles de l'Institit Fourier*, vol. 5, pp. 131–295, 1953.

[33] T. Murofushi and M. Sugeno, "An interpretation of fuzzy measure and the Choquet integral as an integral with respect to a fuzzy measure," *Fuzzy Sets and Systems*, vol. 29, no. 2, pp. 201–227, 1989.

[34] M. Grabisch, "Fuzzy integral in multicriteria decision making," *Fuzzy Sets and Systems*, vol. 69, pp. 279–298, 1995.

[35] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[36] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *ACM Workshop on COLT*, 1992, pp. 144–152.

[37] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philosophical Trans. Royal Society A*, vol. 209, pp. 441–458, 1909.

[38] J. Zhai, H. Xu, and Y. Li, "Fusion of extreme learning machine with fuzzy integral," *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 21, no. 2, pp. 23–34, 2013.

[39] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Sys. Tech.*, vol. 2, no. 3, pp. 1–27, 2011.