Multiple Kernel Fuzzy Clustering

Hsin-Chien Huang, Yung-Yu Chuang, *Member, IEEE*, and Chu-Song Chen, *Member, IEEE*

***Abstract*—While fuzzy c-means is a popular soft-clustering method, its effectiveness is largely limited to spherical clusters. By applying kernel tricks, the kernel fuzzy c-means algorithm attempts to address this problem by mapping data with nonlinear relationships to appropriate feature spaces. Kernel combination, or selection, is crucial for effective kernel clustering. Unfortunately, for most applications, it is uneasy to find the right combination. We propose a multiple kernel fuzzy c-means (MKFC) algorithm that extends the fuzzy c-means algorithm with a multiple kernel-learning setting. By incorporating multiple kernels and automatically adjusting the kernel weights, MKFC is more immune to ineffective kernels and irrelevant features. This makes the choice of kernels less crucial. In addition, we show multiple kernel k-means to be a special case of MKFC. Experiments on both synthetic and real-world data demonstrate the effectiveness of the proposed MKFC algorithm.**

***Index Terms*—Clustering, fuzzy c-means (FCM), multiple kernel learning, soft clustering.**

I. Introduction

CLUSTERING is an unsupervised method to divide data into disjoint subsets with high intracluster similarity and low intercluster similarity. Over the past decades, many clustering algorithms have been proposed, including k-means clustering [1], mixture models [1], spectral clustering [2], locality-sensitive hashing [3], and maximum margin clustering [4], [5]. Most of these approaches perform hard clustering, i.e., they assign each item to a single cluster. This works well when clustering compact and well-separated groups of data, but in many real-world situations, clusters overlap. Thus, for items that belong to two or more clusters, it may be more appropriate to assign them with gradual memberships to avoid coarse-grained assignments of data [6]. This class of clustering methods is called soft- or fuzzy-clustering.

Manuscript received September 16, 2010; revised February 20, 2011 and June 16, 2011; accepted August 15, 2011. Date of publication September 29, 2011; date of current version February 7, 2012. This work was supported in part by the National Science Council of Taiwan under Grant NSC 98-2221-E-001-012-MY3 and Grant NSC100-2622-E-002-016-CC2.

H.-C. Huang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan and also with the Institute of Information Science, Academia Sinica, Taipei, 115, Taiwan (e-mail: sean@cmlab.csie.ntu.edu.tw).

Y.-Y. Chuang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: cyy@csie.ntu.edu.tw).

C.-S. Chen is with the Research Center for Information Technology Innovation and the Institute of Information Science, Academia Sinica, Taipei 115, Taiwan, and also with the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 106, Taiwan (e-mail: song@iis.sinica.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2011.2170175

Fuzzy c-means (FCM) [7], [8] is one of the most promising fuzzy clustering methods. In most cases, it is more flexible than the corresponding hard-clustering algorithms. Unfortunately, as with other clustering methods that are based on the L_2-norm distance in the observation space, it has been shown that while it is effective for spherical clusters, it does not perform well for more general clusters [9]. Thus, kernel-based clustering has been proposed to perform clustering in a typically higher dimensional feature space spanned by embedding maps and corresponding kernel functions [10]. The FCM algorithm has also been extended to the kernel FCM algorithm [11], which yields better performance. However, for such kernel-based methods, a crucial step is the combination or selection of the best kernels among an extensive range of possibilities. This step is often heavily influenced by prior knowledge about the data and by the patterns that we expect to discover [12]. Unfortunately, it is unclear which kernels are more suitable for a particular task [13], [14].

The problem is aggravated for many real-world clustering applications, in which there are multiple potentially useful cues. For such applications, to apply kernel-based clustering, it is often necessary to aggregate features from different sources into a single aggregated feature. However, these features are often not equally relevant to clustering; some are irrelevant, and some are less important than others [9]. As most clustering methods do not embed a feature selection capability, such feature imbalances often necessitate an additional process of feature selection, or feature fusion, before clustering.

Instead of a single fixed kernel, multiple kernels may be used. Recent developments in multiple kernel learning have shown that the construction of a kernel from a number of basis kernels allows for more flexible encoding of domain knowledge from different sources or cues. However, as observed by Zhao *et al.*, previous multiple kernel-learning approaches have focused on supervised and semisupervised learning [13]. A notable exception is their work on multiple kernel maximum margin clustering [13], which is designed for hard clustering.

We here extend the multiple kernel-learning paradigm to fuzzy clustering. The proposed multiple kernel fuzzy c-means (MKFC) algorithm simultaneously finds the best degrees of membership and the optimal kernel weights for a nonnegative combination of a set of kernels. We also embed the feature weight computation into the clustering procedure. The incorporation of multiple kernels and the automatic adjustment of kernel weights renders MKFC more immune to unreliable features or kernels. It also makes combining kernels more practical, since appropriate weights are assigned automatically. Effective kernels or features tend to contribute more to the clustering and, therefore, improve results. Compared with the work of Zhao *et al.* [13], our approach provides the following advantages. First, our method does not require explicit evaluation in the feature space but conducts only kernel-based evaluations.

Thus, our method is more suitable for relational data than their method. Second, MKFC is easy to implement. As mentioned by Zhao *et al.* [13], their formulation leads to a nonconvex integer optimization problem, which is much more difficult to solve. Finally, MKFC yields fuzzy (soft) clustering results that are more appropriate when clusters have significant overlap.

The rest of this paper is organized as follows. In Section II, we discuss related work, and in Section III, we review the FCM algorithm. We derive the MKFC method in Section IV, and we present experiments on both synthetic and real data in Section V. We conclude this paper in Section VI.

II. Related Work

In the FCM algorithm, a data item may belong to more than one cluster with different degrees of membership. The algorithm was first developed by Dunn [7] and was later improved by Bezdek [8]. Dave and Krishnapuram [15] analyzed several popular robust clustering methods and established the connection between fuzzy set theory and robust statistics. Hathaway and Bezdek [16] extended the rough-fuzzy c-means algorithm to arbitrary (non-Euclidean) dissimilarity data. Dave and Sen [17] proposed the fuzzy relational data clustering algorithm that can handle datasets containing outliers and can deal with all kinds of relational data. Parameters such as the fuzzification degree greatly affect the performance of FCM. While Pal and Bezdek [18] suggest that a good setting for this degree for some applications is 2, this choice was based on empirical studies and may not be appropriate for some real datasets. Yu and Yang [19] presented the generalized FCM algorithm, proposing an approach for setting algorithm parameters. Krishnapuram *et al.* [20] presented the fuzzy c-medoid relational clustering algorithm, which can efficiently cluster large datasets.

The use of kernels has received considerable attention because kernels make it possible to map data onto a high-dimensional feature space in order to increase the representation capability of linear machines. Genton [21] presented classes of kernels for machine learning from a statistical perspective. As FCM is similar to the k-means algorithm in that it uses the squared Euclidean distance to measure similarity between prototypes and data points, it is more effective when clustering spherical clusters [11]. Girolami generalized the approach for a wider variety of clusters when he proposed kernel-based clustering [10]. Camastra and Verri [22] presented a kernel-based clustering algorithm that is inspired by the k-means algorithm that iteratively refines results using a one-class support vector machine. Tzortzis and Likas [23] proposed a deterministic and incremental algorithm to overcome the cluster initialization problem: Their algorithm maps data points from the input space to a higher dimensional feature space by the use of a kernel function and optimizes the clustering error. Later, Zhang and Chen proposed the kernel-based fuzzy c-means (KFC) algorithm [11], which allows for incomplete data as well. Shen *et al.* addressed the same problem using weighted KFC for better feature selection [9]. Leski extended the FCM algorithm with insensitivity control so that the resulting method is more robust to noise and outliers [24]. Filippone *et al.* [25] contributed a survey of kernel and spectral clustering methods. Kernel clustering methods are the kernel versions of many classical clustering algorithms such as k-means and self-organized map (SOM). Hathaway *et al.* [26] extended kernelization to relational data clustering by proposing a kernelized form of the non-Euclidean relational FCM algorithm. Chiang and Hao [27] proposed a multiple spheres support vector clustering algorithm based on the adaptive cell growing model which maps data points to a high-dimensional feature space using the desired kernel function. As mentioned by Graves and Pedrycz [14], KFC is divided into two categories. In the first category, prototypes reside in the feature space and are implicitly mapped to the kernel space by the use of a kernel function, whereas in the second category, prototypes are directly constructed in the kernel space, which allows more freedom for prototypes in the feature space.

Our method is related to multiple kernel learning. For kernel methods, the key to success is the formation of a suitable kernel function [13]. However, a single kernel that is selected from a predefined group is sometimes insufficient to represent the data. Different features that are chosen for data can result in different similarity measures corresponding to distinct kernels. The combination of multiple kernels from a set of basis kernels has, therefore, gained acceptance as a way to refine the results of single kernel learning. Multiple kernel learning originates from the work of Lanckriet *et al.* [28], which results in a convex optimization problem for support vector machines. Bach *et al.* suggested an alternative algorithm based on sequential minimization optimization [29]. Efficiency issues of multiple kernel learning were later addressed by Sonnenburg *et al.* using semi-infinite linear programming [30] and by Rakotomamonjy *et al.* using a two-step alternation optimization scheme [31]. Varma and Babu studied superlinear combinations of kernels [32], and Gonen *et al.* studied local combinations of kernels [33]. Frigui and Hwang [34] proposed a semisupervised algorithm that clusters and aggregates relational data (SS-CARD): This algorithm not only partitions the data into meaningful clusters but aggregates pairwise distances from multiple relational matrices and learns a relevance weight for each matrix in each cluster as well. However, most effort along this direction has been spent on supervised learning, in particular, support vector classification and regression. An exception to extend multiple kernels to unsupervised learning, hard clustering, in particular, is the work of Zhao *et al.* [13], which is based on maximum margin clustering. Our work aims to extend multiple kernels to soft clustering.

III. Fuzzy c-Means

In this section, we briefly review the FCM algorithm and its derivation. Given the number of clusters C and a set of data $\mathbf{X}$ containing Nl-dimensional vectors $\mathbf{x}_i$, the FCM algorithm outputs the degrees of membership u_{ic}, i.e., the possibility that data $\mathbf{x}_i$ belong to the c-th cluster, by minimizing the following objective function:

$$J(\mathbf{U},\mathbf{V}) = \sum_{i=1}^{N}\sum_{c=1}^{C} u_{ic}^{m} d^2(\mathbf{x}_i,\mathbf{v}_c)$$

$$\text{subject to } \sum_{c=1}^{C} u_{ic} = 1 \quad \forall i$$

Algorithm 1 Fuzzy c-means (FCM). Given a set of N data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and the desired number of clusters C, output a membership matrix $\mathbf{U} = \{u_{ic}\}_{i,c=1}^{N,C}$.

```
1: procedure FCM(Data $\mathbf{X}$, Number $C$)
2:     Initialize membership matrix $\mathbf{U}^{(0)}$
3:     repeat
4:         update $\mathbf{V}^{(t)} = [\mathbf{v}_c]$ by calculating centers $\mathbf{v}_c$ using
               Equation (4)
5:         update $\mathbf{U}^{(t)} = [u_{ic}]$ by calculating memberships $u_{ic}$
               using Equation (3)
6:     until $\| \mathbf{U}^{(t)} - \mathbf{U}^{(t-1)} \| < \epsilon$
7:     return $\mathbf{U}^{(t)}$
8: end procedure
```

$$\text{and } u_{ic} \geq 0 \quad \forall i, c$$

$$\text{and } \sum_{i=1}^{N} u_{ic} > 0 \quad \forall c \tag{1}$$

where m is the fuzzification degree, which should be larger than 1; $d(\cdot, \cdot)$ is the Euclidean distance; $\mathbf{v}_c$ is the center of the cth cluster; $\mathbf{U} = [u_{ic}]_{i=1..N, c=1..C}$ is an $N \times C$ membership matrix whose elements are the degrees of membership; and $\mathbf{V} = [\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_C]$ is an $l \times C$ matrix whose columns correspond to cluster centers. In the FCM algorithm, we solve the previous constrained optimization problem using Lagrange multipliers:

$$J_\lambda(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{N} \sum_{c=1}^{C} u_{ic}^m d^2(\mathbf{x}_i, \mathbf{v}_c) + \lambda \left(\sum_{c=1}^{C} u_{ic} - 1 \right). \tag{2}$$

The problem is solved by iteratively updating degrees of membership with fixed centers and updating centers with fixed degrees of membership. The closed-form formulas for updates are derived by taking the partial derivatives with respect to both and setting them to zero:

$$u_{ic} = \frac{1}{\sum_{c'=1}^{C} \left(\frac{d(\mathbf{x}_i, \mathbf{v}_c)}{d(\mathbf{x}_i, \mathbf{v}_{c'})} \right)^{\frac{2}{m-1}}} \tag{3}$$

and

$$\mathbf{v}_c = \frac{\sum_{i=1}^{N} u_{ic}^m \mathbf{x}_i}{\sum_{i=1}^{N} u_{ic}^m}. \tag{4}$$

One thing to note is that although we do not add any Lagrange multiplier for the nonnegative constraints in (1), it can be verified that the aforementioned formula implicitly satisfies constraints such as $u_{ic} \geq 0 \, \forall i, c$. In addition, when m is very close to 1, the FCM algorithm degenerates to the k-means algorithm. Algorithm 1 summarizes the FCM algorithm.

IV. Multiple Kernel Fuzzy c-Means

A. Objective Function

To discover nonlinear relationships among data, kernel methods use embedding mappings that map features of the data to new feature spaces [12]. Consider a set of M such mappings, i.e., $\Phi = \{\phi_1, \phi_2, \cdots, \phi_M\}$. Each mapping ϕ_k recodes the l-d data $\mathbf{x}$ as a vector $\phi_k(\mathbf{x})$ in its feature space whose dimensionality is L_k. Let $\{\kappa_1, \kappa_2, \ldots, \kappa_M\}$ be the Mercer kernels corresponding to these implicit mappings, respectively

$$\kappa_k(\mathbf{x}_i, \mathbf{x}_j) = \phi_k(\mathbf{x}_i)^T \phi_k(\mathbf{x}_j).$$

To combine these kernels, as well as to ensure that the resulted kernel still satisfies Mercer's condition, we consider a nonnegative combination of these feature maps, ϕ', i.e.,

$$\phi'(\mathbf{x}) = \sum_{k=1}^{M} \omega_k \phi_k(\mathbf{x}), \text{with } \omega_k \geq 0.$$

Unfortunately, as these implicit mappings do not necessarily have the same dimensionality, such a linear combination may be impossible. Hence, we construct a new set of independent mappings $\Psi = \{\psi_1, \psi_2, \ldots, \psi_M\}$ from the original mappings Φ as

$$\psi_1(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \psi_2(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \phi_2(\mathbf{x}) \\ \vdots \\ \mathbf{0} \end{bmatrix}, \ldots, \psi_M(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}.$$

Each of these constructed mappings converts $\mathbf{x}$ into an L-d vector, where $L = \sum_{k=1}^{M} L_k$. Note that it is possible that some feature spaces have infinite dimensionalities. In such cases, we can always interlace the dimensions of these features so that they still form a set of orthogonal bases. However, as we later eliminate evaluation in the feature space, this will not matter. To construct new mappings this way ensures that the feature spaces of these mappings have the same dimensionality, and their linear combination can be well defined. In addition, these mappings form a new set of orthogonal bases since

$$\psi_k(\mathbf{x}_i)^T \psi_k(\mathbf{x}_j) = \kappa_k(\mathbf{x}_i, \mathbf{x}_j)$$
$$\psi_k(\mathbf{x}_i)^T \psi_{k'}(\mathbf{x}_j) = 0, \quad \text{if} \quad k \neq k'.$$

As such orthogonal bases prevent cross terms between different implicit mappings, we can focus on the inner product of data of the same mapping that can be well evaluated by the original kernel functions. We seek to find $\psi(\mathbf{x}) = \sum_{k=1}^{M} \omega_k \psi_k(\mathbf{x})$, i.e., a nonnegative linear expansion of the bases in Ψ, which maps data to an implicit feature space. Thus, the objective function becomes

$$J(\mathbf{w}, \mathbf{U}, \mathbf{V}) = \sum_{i=1}^{N} \sum_{c=1}^{C} u_{ic}^m \left(\psi(\mathbf{x}_i) - \mathbf{v}_c \right)^T \left(\psi(\mathbf{x}_i) - \mathbf{v}_c \right) \tag{5}$$

$$\psi(\mathbf{x}) = \omega_1 \psi_1(\mathbf{x}) + \omega_2 \psi_2(\mathbf{x}) + \cdots + \omega_M \psi_M(\mathbf{x})$$

$$\text{subject to } \omega_1 + \omega_2 + \cdots + \omega_M = 1 \tag{6}$$

$$\text{and } \omega_k \geq 0 \quad \forall k \tag{7}$$

$$\text{and } \sum_{c=1}^{C} u_{ic} = 1 \quad \forall i \tag{8}$$

$$\text{and } u_{ic} \geq 0 \quad \forall i, c$$

$$\text{and } \sum_{i=1}^{N} u_{ic} > 0 \quad \forall c \tag{9}$$

where $\mathbf{v}_c$ is the center of the cth cluster in the implicit feature space, $\mathbf{w} = (\omega_1, \omega_2, \ldots, \omega_M)^T$ is a vector consisting of weights, $\mathbf{U}$ is an $N \times C$ membership matrix whose elements are the memberships u_{ic}, and $\mathbf{V}$ is an $L \times C$ matrix whose columns correspond to cluster centers.

B. Optimizing Memberships

The goal of MKFC is to simultaneously find combination weights $\mathbf{w}$, memberships $\mathbf{U}$, and cluster centers $\mathbf{V}$, which minimize the objective function in (5). However, direct evaluation of the cluster centers may not be possible because they are in the implicit feature space; we show later that their associated computation can be replaced by the kernel trick. Similar to FCM, we first fix the weights and cluster centers to find the optimal memberships. For brevity, we use D_{ic} to denote the distance between data $\mathbf{x}_i$ and cluster center $\mathbf{v}_c$, i.e., $D_{ic}^2 = (\psi(\mathbf{x}_i) - \mathbf{v}_c)^T\, (\psi(\mathbf{x}_i) - \mathbf{v}_c)$. Thus, (5) can be written as

$$J(\mathbf{w}, \mathbf{U}, \mathbf{V}) = \sum_{i=1}^{N} \sum_{c=1}^{C} u_{ic}^m D_{ic}^2. \tag{10}$$

When the weights and cluster centers are fixed, the distances are constants. Similar to FCM [see (2)], by forming an energy function with Lagrange multiplier λ for the constraint $\sum_{c=1}^{C} u_{ic} = 1$, we have

$$J_\lambda(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{N} \sum_{c=1}^{C} u_{ic}^m D_{ic}^2 + \lambda \left(\sum_{c=1}^{C} u_{ic} - 1 \right).$$

Next, we take its derivatives with respect to the memberships and set them to zero; for each membership u_{ic}, we have

$$\frac{\partial J_\lambda}{\partial u_{ic}} = m D_{ic}^2 u_{ic}^{m-1} + \lambda = 0.$$

The solution for u_{ic} is

$$u_{ic} = \left(\frac{-\lambda}{m} \right)^{\frac{1}{m-1}} \frac{1}{D_{ic}^{2/(m-1)}}.$$

Because of the constraint $\sum_{c=1}^{C} u_{ic} = 1$, we can eliminate λ and obtain the closed-form solution for the optimal memberships as

$$u_{ic} = \frac{1}{\sum_{c'=1}^{C} \left(\frac{D_{ic}^2}{D_{ic'}^2} \right)^{\frac{1}{m-1}}}. \tag{11}$$

C. Optimizing Weights

From (11), it can be seen that when the weights $\mathbf{w}$ and cluster centers $\mathbf{V}$ are fixed, the optimal memberships $\mathbf{U}$ can be obtained. Now, let us assume that the memberships are fixed. We seek to derive the optimal centers and weights to combine the kernels. By taking the derivative of $J(\mathbf{w}, \mathbf{U}, \mathbf{V})$ in (5) with respect to $\mathbf{v}_c$ and setting it to zero, we have

$$\frac{\partial J(\mathbf{w}, \mathbf{U}, \mathbf{V})}{\partial \mathbf{v}_c} = -2 \sum_{i=1}^{N} u_{ic}^m (\psi(\mathbf{x}_i) - \mathbf{v}_c) = 0.$$

Hence, when $\mathbf{U}$ are given, the optimal $\mathbf{v}_c$ is the following closed-form solution represented by the combination weights

$$\mathbf{v}_c = \frac{\sum_{i=1}^{N} u_{ic}^m \psi(\mathbf{x}_i)}{\sum_{i=1}^{N} u_{ic}^m} = \sum_{i=1}^{N} \hat{u}_{ic} \psi(\mathbf{x}_i) \tag{12}$$

where $\hat{u}_{ic} = \frac{u_{ic}^m}{\sum_{i=1}^{N} u_{ic}^m}$ is the normalized membership. However, these cluster centers are in the kernel-defined feature space which might be implicit or even have an infinite dimensionality. Therefore, it may be impossible to directly evaluate these centers. Fortunately, for clustering, it is often sufficient to just obtain the memberships; we later show that it is possible to obtain memberships and weights without implicitly evaluating cluster centers. Thus, we focus on finding optimal weights for fixed memberships when the cluster centers are the closed-form optimal solution [see (12)]

$$\begin{aligned} D_{ic}^2 &= (\psi(\mathbf{x}_i) - \mathbf{v}_c)^T\, (\psi(\mathbf{x}_i) - \mathbf{v}_c) \\ &= \psi(\mathbf{x}_i)^T \psi(\mathbf{x}_i) - 2\psi(\mathbf{x}_i)^T \left(\sum_{j=1}^{N} \hat{u}_{jc} \psi(\mathbf{x}_j) \right) \\ &\quad + \left(\sum_{j=1}^{N} \hat{u}_{jc} \psi(\mathbf{x}_j) \right)^T \left(\sum_{j'=1}^{N} \hat{u}_{j'c} \psi(\mathbf{x}_{j'}) \right) \\ &= \sum_{k=1}^{M} \omega_k^2 \kappa_k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{j=1}^{N} \sum_{k=1}^{M} \hat{u}_{jc} \omega_k^2 \kappa_k(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \sum_{j=1}^{N} \sum_{j'=1}^{N} \sum_{k=1}^{M} \hat{u}_{jc} \hat{u}_{j'c} \omega_k^2 \kappa_k(\mathbf{x}_j, \mathbf{x}_{j'}). \end{aligned} \tag{13}$$

Since memberships are fixed and kernel functions can be evaluated, (13) can be rearranged as

$$D_{ic}^2 = \sum_{k=1}^{M} \alpha_{ick} \omega_k^2 \tag{14}$$

where the coefficient α_{ick} can be written as

$$\begin{aligned} \alpha_{ick} =& \kappa_k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{j=1}^{N} \hat{u}_{jc} \kappa_k(\mathbf{x}_i, \mathbf{x}_j) \\ &+ \sum_{j=1}^{N} \sum_{j'=1}^{N} \hat{u}_{jc} \hat{u}_{j'c} \kappa_k(\mathbf{x}_j, \mathbf{x}_{j'}). \end{aligned} \tag{15}$$

Note that we have eliminated cluster centers from the evaluation. Thus, the objective function in (10) becomes

$$\begin{aligned} J(\mathbf{w}, \mathbf{U}) = & \sum_{i=1}^{N} \sum_{c=1}^{C} u_{ic}^m \sum_{k=1}^{M} \alpha_{ick} \omega_k^2 \\ & \text{subject to } \omega_1 + \omega_2 + \cdots + \omega_M = 1 \\ & \text{and } \omega_k \geq 0 \quad \forall k \\ & \text{and } \sum_{c=1}^{C} u_{ic} = 1 \quad \forall i \\ & \text{and } u_{ic} \geq 0 \quad \forall i, c. \end{aligned} \tag{16}$$

When memberships are fixed, we have

$$J(\mathbf{w}) = \sum_{k=1}^{M} \beta_k \omega_k^2$$
$$\text{subject to } \omega_1 + \omega_2 + \cdots + \omega_M = 1$$
$$\text{and } \omega_k \geq 0 \quad \forall k \tag{17}$$

where the coefficient β_k is

$$\beta_k = \sum_{i=1}^{N}\sum_{c=1}^{C} u_{ic}^m \alpha_{ick}. \tag{18}$$

This is a constrained optimization problem. By introducing a Lagrange multiplier, we have

$$J_\lambda(\mathbf{w}, \lambda) = \sum_{k=1}^{M} \beta_k \omega_k^2 - 2\lambda \left(\sum_{k=1}^{M} \omega_k - 1\right).$$

Note that for now, we ignore the constraint that weights must be nonnegative. Later, we show that it is satisfied in our solution. By taking the partial derivatives and setting them to zero, we have

$$\frac{\partial J_\lambda}{\partial \omega_k} = 2\beta_k\omega_k - 2\lambda = 0.$$

The solution for the aforementioned equation is $\omega_k = \frac{\lambda}{\beta_k}$. In addition, we know that

$$\sum_{k=1}^{M} \omega_k = \left(\frac{1}{\beta_1} + \frac{1}{\beta_2} + \cdots + \frac{1}{\beta_M}\right)\lambda = 1.$$

Thus, we have

$$\lambda = \frac{1}{\frac{1}{\beta_1} + \frac{1}{\beta_2} + \cdots + \frac{1}{\beta_M}}$$

and the weight is the harmonic mean

$$\omega_k = \frac{\frac{1}{\beta_k}}{\frac{1}{\beta_1} + \frac{1}{\beta_2} + \cdots + \frac{1}{\beta_M}}. \tag{19}$$

Previously, we have derived the alternative optimizations of memberships and kernel combination weights. However, the derivations are based only on the equality constraints [see (6) and (8)] and do not take into account the inequality constraint, i.e., the memberships and weights should not be negative [see (7) and (9)]. Since it is easy to verify that the derived memberships [see (11)] always satisfy $u_{ic} \geq 0$ and $\sum_{i=1}^{N} u_{ic} > 0\,\forall c$, we show that the solution of the combination weights also satisfies the non-negative constraint, i.e., $\omega_k \geq 0$. We first show that $\beta_k \geq 0$ for all k. By definition, D_{ic}^2 should always be nonnegative for all weights, i.e., $\forall\omega_k, D_{ic}^2 = \sum_{k=1}^{M} \alpha_{ick}\omega_k^2 \geq 0$. Thus, we can conclude that $\forall\omega_k, \alpha_{ick} \geq 0$. Otherwise, if $\alpha_{ick'} < 0$ for some k', we can let $\omega_{k'} = 1$ and $\omega_k = 0$ if $k \neq k'$. However, this set of weight assignments means that $D_{ic}^2 < 0$, which contradicts its nonnegative property. Therefore, since both α_{ick} and u_{ic} are nonnegative, from (18), we conclude that $\beta_k \geq 0$. Finally, since ω_k's are harmonic means of nonnegative β_k's as shown in (19), they are also nonnegative. Thus, even though we do not initially take into account the nonnegative constraint, the solution that we obtain automatically satisfies this constraint.

Algorithm 2 Multiple kernel fuzzy c-means (MKFC). Given a set of N data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, a set of kernel functions $\{\kappa_k\}_{k=1}^M$, and the desired number of clusters C, output a membership matrix $\mathbf{U} = \{u_{ic}\}_{i,c=1}^{N,C}$ and weights $\{\omega_k\}_{k=1}^M$ for the kernels.

```
1: procedure MKFC(Data $\mathbf{X}$, Number $C$, Kernels $\{\kappa_k\}_{k=1}^M$)
2:     Initialize membership matrix $\mathbf{U}^{(0)}$
3:     repeat
4:         $\hat{u}_{ic}^{(t)} = \frac{u_{ic}^{(t)m}}{\sum_{i=1}^N u_{ic}^{(t)m}}$   ▷ calculate normalized memberships
                                   ▷ calculate coefficients by Equation (15)
5:         for ($i = 1..N; c = 1..C; k = 1..M$) do
6:             $\alpha_{ick} \leftarrow \kappa_k(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^N \hat{u}_{jc}^{(t)}\kappa_k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{j=1}^N\sum_{j'=1}^N \hat{u}_{jc}^{(t)}\hat{u}_{j'c}^{(t)}\kappa_k(\mathbf{x}_j, \mathbf{x}_{j'})$
7:         end for
                                   ▷ calculate coefficients by Equation (18)
8:         for ($k = 1..M$) do
9:             $\beta_k \leftarrow \sum_{i=1}^N \sum_{c=1}^C \left(u_{ic}^{(t)}\right)^m \alpha_{ick}$
10:        end for
                                   ▷ update weights by Equation (19)
11:        for ($k = 1..M$) do
12:            $\omega_k^{(t)} \leftarrow \frac{\frac{1}{\beta_k}}{\frac{1}{\beta_1}+\frac{1}{\beta_2}+\cdots+\frac{1}{\beta_M}}$
13:        end for
                                   ▷ calculate distances by Equation (14)
14:        for ($i = 1..N; c = 1..C$) do
15:            $D_{ic}^2 \leftarrow \sum_{k=1}^M \alpha_{ick}\left(\omega_k^{(t)}\right)^2$
16:        end for
                                   ▷ update memberships by Equation (11)
17:        for ($i = 1..N; c = 1..C$) do
18:            $u_{ic}^{(t)} \leftarrow \frac{1}{\sum_{c'=1}^C \left(\frac{D_{ic}^2}{D^2{ic'}}\right)^{\frac{1}{m-1}}}$
19:        end for
20:    until $\parallel \mathbf{U}^{(t)} - \mathbf{U}^{(t-1)} \parallel < \epsilon$
21:    return $\mathbf{U}^{(t)}, \{\omega_k^{(t)}\}_{k=1}^M$
22: end procedure
```

D. Algorithm

We start from the objective function involving cluster centers, memberships, and kernel weights. We show that the cluster centers can be eliminated from the objective function so that we do not need to implicitly evaluate cluster centers, which are potentially not computable. Algorithm 2 summarizes the MKFC algorithm, which starts by initializing a random membership matrix satisfying nonnegative and unity constraints. Optimal weights are calculated by fixing the memberships, and optimal memberships are updated assuming fixed weights. The process

TABLE I
SUMMARY OF THE DATASETS USED IN THE EXPERIMENTS

ID	Name	#instances	#features	#classes	comment
S1	*Equal*	160	10	8	
S2	*Variant*	160	10	8	
R1	*Wine*	178	13	3	
R2	*Glass Identification*	214	9	6	
R3	*SPECT Heart*	267	22	2	
R4	*Ecoli*	336	7	8	
R5	*Ionosphere*	351	34	2	
R6	*Libras Movement*	360	90	15	
R7	*Breast Cancer Wisconsin(Diagnostic)*	569	30	2	
R8	*Balance Scale*	625	4	3	
R9	*Optical Recognition of Handwritten Digits*$(1, 7)$	361	64	2	digit 1 and 7 in test set
R10	*Optical Recognition of Handwritten Digits*$(2, 7)$	356	64	2	digit 2 and 7 in test set
R11	*Optical Recognition of Handwritten Digits*$(0, 6, 8, 9)$	713	64	4	digit 0,6,8 and 9 in test set
R12	*Optical Recognition of Handwritten Digits*$(1, 2, 7, 9)$	718	64	4	digit 1,2,7 and 9 in test set
R13	*Pima Indians Diabetes*	768	8	2	
R14	*Connectionist Bench(Vowel Recognition-Deterding Data)*	990	10	11	
R15	*Yeast*	1,484	8	10	
R16	*Statlog(Landsat Satellite)*	2,236	36	2	class 1 and 2
R17	*Statlog(Landsat Satellite)*	480	36	6	randomly choose 80 instances from each class
R18	*Letter Recognition*(A,B)	1,555	16	2	letter A and B
R19	*Letter Recognition*(A,B,C,D)	3,096	16	4	letter A,B,C,D
R20	*Waveform Database Generator(Version 2)*	5,000	21	3	
F1	*ORL*	360	7744	40	
F2	*CMU-PIE*	1496	7744	68	The frontal images (Pose 27) with 22 different lightings
T1	*20 Newsgroups*	2000	25753	20	randomly choose 100 instances from each class in test set
T2	*Reuters-21578*	2189	8575	8	The test set

The first two, *Equal* and *Variant*, are synthetic datasets whose dimensions have different feature reliability characteristics. The following 20 datasets are adopted from the UCI machine learning repository, two face databases from ORL and CMU-PIE, and the last two are text datasets from 20 Newsgroups and Reuters-21578.

is repeated until the amount of change per iteration in the membership matrix falls below a given threshold. The computational complexity of MKFC is $O(N^2CM)$ per iteration, excluding construction of the kernel matrices.

E. Multiple Kernel K-Means: A Special Case of Multiple Kernel Fuzzy c-Means

The proposed MKFC method can be viewed as a multiple kernel extension of k-means, which is the most widely used clustering algorithm. The argument is similar to that for k-means being a special case of FCM.

Consider (11) when m approaches 1. If there exists any cluster c' such that $D_{ic'} < D_{ic}$, then the denominator approaches ∞. On the other hand, if D_{ic} is the smallest among all $D_{ic'}$, the denominator approaches zero, i.e.,

$$u_{ic} = \begin{cases} 0, & \text{if there exists } c' \text{ such that } D_{ic'} < D_{ic} \\ 1, & \text{otherwise.} \end{cases}$$

Since u_{ic} is either 1 or 0 as determined by the nearest center, MKFC reduces to hard clustering. We here refer to the resultant hard-clustering version of MKFC (when m is very close to 1) as multiple kernel k-means (MKKM). MKKM is depicted in Algorithm 3. To the best of our knowledge, there is no previous study that has extended k-means to MKKM.

V. EXPERIMENTS

We begin this section by reviewing the measures that we have adopted to evaluate and compare the clustering results (see Section V-A), after which we discuss the issue of the base kernel selection (see Section V-B). We conclude the section with a presentation of experiments on synthetic data (see Section V-C), a number of real datasets from the University of California, Irvine (UCI) machine learning repository [35] (see Section V-D), two well-known face databases from ORL [36] and CMU-PIE [37] (see Section V-E), and two well-known text datasets from 20 Newsgroups and Reuters-21578 (see Section V-F).

For each set of experiments, we describe the datasets, the experimental settings, the choice of kernels, the experimental results, and comparisons with other methods.

These datasets are summarized in Table I. For all experiments, we set the fuzzification degree m to 1.08 and the stop threshold ϵ as 0.0001. Since the focus of this paper is not the estimation of the number of clusters, we set as the ground truth for all methods the number of clusters C. Because the performance of these clustering methods depends on the initial values, we performed 50 runs for each experiment and report the average. We compare the proposed method (MKFC) and MKKM with k-means (Kmean), FCM, and single-KFC.

A. Performance Measures

The FCM-based soft-clustering algorithms (i.e., FCM, KFC, and MKFC) that are described in this paper generate an $N \times C$

matrix $\mathbf{U} = [u_{ic}]_{i=1..N,c=1..C}$ whose elements $u_{ic} \in [0,1]$ are the membership degrees, the possibility that data $\mathbf{x}_i$ belongs to the cth cluster. These membership degrees make it possible for us to measure the performance of these algorithms using either hard-clustering measures or soft-clustering measures.

Hard-Clustering Measures: Most clustering measures are designed for the evaluation of the results of hard clustering, in which each data item is assigned to a single class. To use this kind of measure for soft clustering, one must convert the membership degrees to hard assignments. We take the conventional approach for such assignments, i.e., we assign each data item to the cluster with the highest membership degree. Hard-clusterings measures can be roughly categorized into pair-counting-based measures (e.g., Rand index (RI) and adjusted Rand index (ARI) [38]), set-matching-based measures (e.g., $\mathcal{H}$ criterion), and information-theoretic-based measures (e.g., mutual information and normalized mutual information (NMI) [39]).

While there are studies that evaluate these clustering measures, there is currently no definite answer as to which measure is best. Vinh *et al.* [40] reported that some popular measures do not facilitate informative clustering comparisons because they either do not have a predetermined range or do not have a constant baseline value. For these measures, a poor clustering could yield a very high performance index, especially with a large number of clusters. They suggest the ARI as a faithful measure that does not have these drawbacks. However, Wu *et al.* [41] reported that when clustering performances are hard to distinguish, we may still want to use the normalized variation of information, i.e., NMI. For fair comparisons, this paper uses both NMI and ARI as hard-clustering measures.

The goal of NMI [39] is to compare two hard partitions (R and Q) of a dataset $\mathbf{X}$ with N objects. Assume that R and Q have I and J clusters, respectively. The probability $P(i)$ that a randomly selected object from $\mathbf{X}$ falls into cluster R_i of partition R is

$$P(i) = \frac{|R_i|}{N}.$$

The entropy $H(R)$ associated with R is then defined as

$$H(R) = -\sum_{i=1}^{I} P(i) \log P(i).$$

Let $P(i,j)$ denote the probability that an object belongs to cluster R_i in R and cluster Q_j in Q:

$$P(i,j) = \frac{|R_i \cap Q_j|}{N}.$$

The NMI between the two hard partitions R and Q can then be defined as

$$\mathrm{NMI}(R,Q) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} P(i,j) \log \frac{P(i,j)}{P(i)P(j)}}{\sqrt{H(R)\,H(Q)}}. \tag{20}$$

In describing the formula for ARI, we start with the definitions of the following quantities:

- a number of pairs of data objects belonging to the same class in R and to the same cluster in Q;
- b number of pairs of data objects belonging to the same class in R and to different clusters in Q;
- c number of pairs of data objects belonging to different classes in R and to the same cluster in Q;
- d number of pairs of data objects belonging to different classes in R and to different clusters in Q.

The RI is then defined as

$$\mathrm{RI} = \frac{a+d}{a+b+c+d}$$

and the ARI is

$$\mathrm{ARI} = \frac{a - \frac{(a+b)(a+c)}{a+b+c+d}}{\frac{(a+b)+(a+c)}{2} - \frac{(a+b)(a+c)}{a+b+c+d}}. \tag{21}$$

By measuring the ARI between the clustering results and the ground-truth clustering, we can evaluate the clustering performance for each method.

Soft-Clustering Measures: As pointed out by Campello [42], the casting of soft clusterings to hard clusterings often fails to faithfully reflect the performance of soft-clustering algorithms. For example, different fuzzy partitions (with potentially widely divergent spatial distributions) may result in the same crisp partition; accordingly, both will have the same hard-clustering measure. This loss of information, caused by the disposal of the fuzzy membership values, makes the hard clustering measures unable to discriminate between overlapped and nonoverlapped clusters. As such, these hard-clustering measures might not be appropriate for the assessment of fuzzy clustering algorithms. To get around these drawbacks, Campello proposed a fuzzy extension of the RI and other related indexes [42]. The extended index is obtained by first rewriting the formulation of the RI in a fully equivalent form using basic concepts from set theory. Given two membership matrices (U_1 and U_2), the quantities $a, b, c,$ and d are redefined as follows when r and q are two soft partitions:

$$\begin{aligned} a &= |V \cap Y| \\ b &= |V \cap Z| \\ c &= |X \cap Y| \\ d &= |X \cap Z| \end{aligned} \tag{22}$$

where V, which is defined as $V(j_1, j_2) = s_{i=1}^{k} t(r_{ij_1}, r_{ij_2})$, is the set of pairs of data objects belonging to the same class in U_1; X, which is defined as $X(j_1, j_2) = s_{i_1, i_2 \in [1,k] | i_1 \neq i_2} t(r_{i_1 j_1}, r_{i_2 j_2})$, is the set of pairs of data objects belonging to different classes in U_1; Y, which is defined as $Y(j_1, j_2) = s_{l=1}^{v} t(q_{lj_1}, q_{lj_2})$, is the set of pairs of data objects belonging to the same cluster in U_2; and Z, which is defined as $Z(j_1, j_2) = s_{l_1, l_2 \in [1,v] | l_1 \neq l_2} t(q_{l_1 j_1}, q_{l_2 j_2})$, is the set of pairs of data objects belonging to different clusters in U_2.

Here, "t" is a t-norm that is used as a conjunction to implement the connective "and" of the proposition: We use the "min" operator as a t-norm. Likewise, "s" is a co-norm that is used as a disjunction to implement the connective "or" of the proposition: We use the "max" operator as a co-norm. Since the cardinality of a fuzzy set is given by the sum of its membership values, we rewrite (22) as

$$a = |V \cap Y| = \sum_{j_2=2}^{N} \sum_{j_1=1}^{j_2-1} t\left(V\left(j_1, j_2\right), Y\left(j_1, j_2\right)\right)$$

$$b = |V \cap Z| = \sum_{j_2=2}^{N} \sum_{j_1=1}^{j_2-1} t\left(V\left(j_1, j_2\right), Z\left(j_1, j_2\right)\right)$$

$$c = |X \cap Y| = \sum_{j_2=2}^{N} \sum_{j_1=1}^{j_2-1} t\left(X\left(j_1, j_2\right), Y\left(j_1, j_2\right)\right)$$

$$d = |X \cap Z| = \sum_{j_2=2}^{N} \sum_{j_1=1}^{j_2-1} t\left(X\left(j_1, j_2\right), Z\left(j_1, j_2\right)\right).$$

Plugging the aforementioned quantities into (21) yields extended ARI (EARI), i.e., the fuzzy extension to ARI. As we are not aware of any soft extension for NMI, for the soft-clustering measure, we use only EARI. In the following experiments, we use NMI, ARI, and EARI to compare algorithms.

B. Selection of Base Kernels

Kernels are often used to address the problems of ineffective features and similarity measures. Features can be ineffective for two possible reasons. First, the data could exhibit nonlinear relationships. For better modeling, kernel functions define the similarity of data in a more appropriate feature space. Second, the provided feature vectors may not faithfully reflect the intrinsic properties of the data, and thus, the resulting similarities will not reflect the actual similarities between data items.

As kernel functions are essentially similarity measures for pairs of data, they can be used in many different ways, and sets of multiple kernels can be constructed in various ways. There are two common ways to construct kernel functions, corresponding to the two situations mentioned previously. First, given a set of representative vectors for data items, one can employ a number of reproducible kernel functions in the Hilbert space for the construction of multiple kernels. For example, we could measure the similarities between data items in different nonlinear spaces by mapping data to these spaces with different Gaussian kernels. For the UCI experiment, since we did not have access to the raw data, we constructed kernels this way. Second, given a set of raw data, different types of feature vectors can be extracted. These feature vectors often correspond to different cues and similarities can be measured in different feature spaces. For example, given a set of facial images for face clustering, one could extract different types of visual features such as colors or textures. For the face-clustering experiment, we extract three types of features to define different similarity measures (kernels). For the synthetic data, to facilitate analysis, we treat each dimension as a feature. This allows for intuitive interpretation of the experimental results.

In summary, the guideline to select base kernels is to use kernels (features) that are known to be effective in related problems. For example, Gaussian kernels are known to be effective in many classification problems, and local binary pattern (LBP) is a popular feature in face image applications. In principle, the more the kernels (features) are used, the better the performance will be. We are still, however, limited by computational resources and the algorithm's tolerance to bad kernels.

C. Synthetic Data

We first evaluate MKFC on synthetic data because we know the correct answers for this case. Our primary goal is to show that because the proposed method assigns proper weights to kernels, it is less vulnerable to irrelevant or ineffective features or kernels. For this purpose, we synthesize l-dimensional vectors and treat each dimension as a feature. Given a pair of data $\mathbf{x}_i$, $\mathbf{x}_j$, which are l-dimensional vectors, each of its dimensions $\mathbf{x}_{ir(r=1\cdots l)}$ is taken as a scalar feature, and a kernel function $\kappa(\cdot,\cdot)$ in 1-D space is used to measure the similarity between $\mathbf{x}_{ir}$ and $\mathbf{x}_{jr}$. In this way, l kernels (similarities) can be obtained for each of the data pair $\mathbf{x}_i$, $\mathbf{x}_j$. Two sets of synthetic data were generated as follows. For the first one, *Equal*, we synthesized eight groups of 10-D data, with 20 data points in each group. We made these groups well separated by sampling eight points, whose coordinates are uniformly sampled along each dimension. These eight points were then used as centers of Gaussians with the same width. For each Gaussian, 20 points were drawn. For this setup, it is reasonable to assume that each dimension has a roughly equal capability to separate the eight groups. A similar procedure was used for the second set, *Variant*. The only difference lies in the widths of the Gaussians. In *Equal*, we used the same width in each dimension, but in *Variant*, the widths were increased with the dimension index. Therefore, there was more overlap in *Variant* data in the dimensions with higher indexes.

Accordingly, we use a Gaussian kernel for each dimension:

$$\kappa_r(\mathbf{x}_{ir}, \mathbf{x}_{jr}) = \exp(-\|\mathbf{x}_{ir} - \mathbf{x}_{jr}\|^2/\sigma).$$

To choose σ, let the minimal value allowed for the Gaussian kernel over the dataset be γ. We then obtain the corresponding σ as

$$\sigma = \min_{i,j}(-\|\mathbf{x}_{ir} - \mathbf{x}_{jr}\|^2/\log(\gamma)).$$

We set γ to 0.005. Fig. 1(a) illustrates the results for the *Equal* experiment. The x-axis is for the NMI index values. Each row shows the NMI value distribution over runs for a method. From left to right, the three green points of each row are sequentially the minimal, mean, and maximal NMI values among the runs. Thirteen methods are compared: k-means (Kmean), FCM, MKFC, and ten KFC methods. In the synthetic-data experiment, k-means and FCM use ten-dimensional feature vectors, but each of the KFC methods uses only one dimension as the feature. Hence, we expect k-means and FCM to outperform each KFC, since the latter method uses incomplete data. As our MKFC method uses a weighted combination of these 1-D kernels, we expect it to outperform single kernel-based methods. As can be seen, although the maximal NMI value of MKFC is the same as for Kmean and FCM, MKFC has a better mean NMI index (i.e., 0.9521) than the other two (i.e., 0.9486 for FCM and 0.9206 for Kmean). Note that we expect MKFC to exhibit performance similar to that of FCM, since the features are equally important in this synthetic dataset. The real

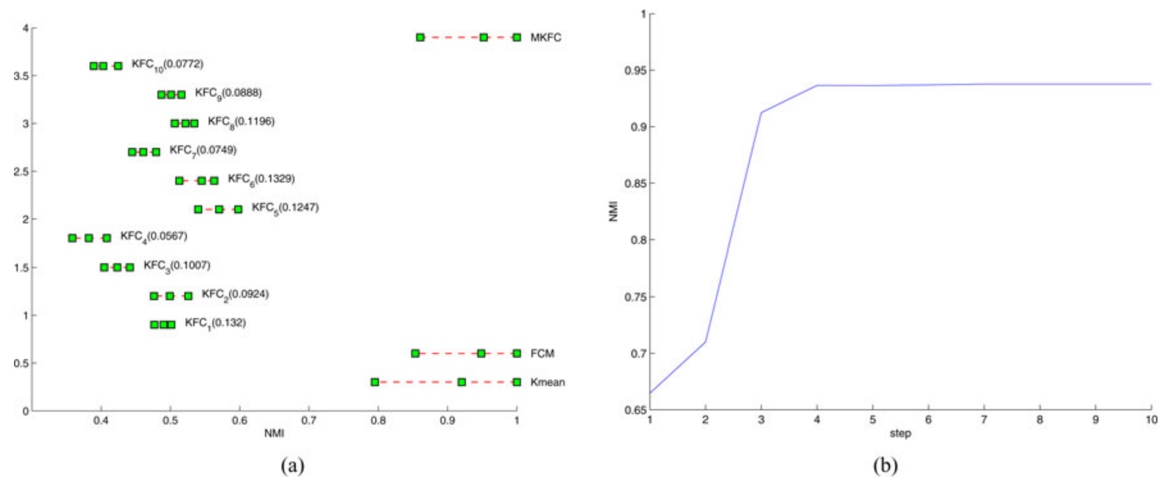


Fig. 1. For the *Equal* experiment, (a) the results and (b) the NMI values for each step of a single run.

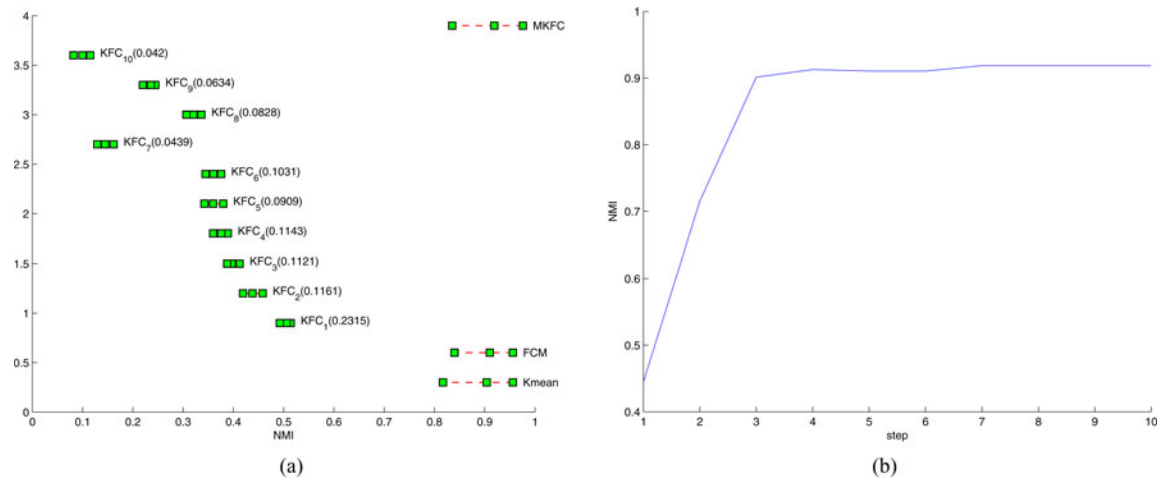


Fig. 2. For the *Variant* experiment, (a) the results and (b) the NMI values for each step of a single run.

numbers for each KFC method are the weights that are discovered by MKFC. These kernel weights are all of similar magnitudes, which is appropriate, considering the artificial setting of equal weights in this synthetic dataset. Note that because of the overlap between clusters and noise, the discovered weights are not perfectly equal. Fig. 1(b) shows the NMI values for each MKFC iteration for the *Equal* experiment: It converges after only a few iterations.

Fig. 2 shows the results for the *Variant* experiment. As in the *Equal* experiment, k-means and FCM use ten-dimensional vectors, each KFC uses only one dimension as the feature, and MKFC uses a weighted combination of these ten dimensions. Because of the experiment settings, there is more data overlap in the dimensions of higher indexes, i.e., for data in these dimensions, it is more difficult to separate the eight groups. This is evident from Fig. 2(a). We observe that the NMI value decreases as the index of KFC increases. Accordingly, MKFC assigns higher weights to the kernels corresponding to the dimensions with lower indices. For example, the weight for KFC$_1$ is 0.23147, while the weight for KFC$_{10}$ is only 0.041994. Fig. 3 shows the evolution of weights in a run. In the first iteration,

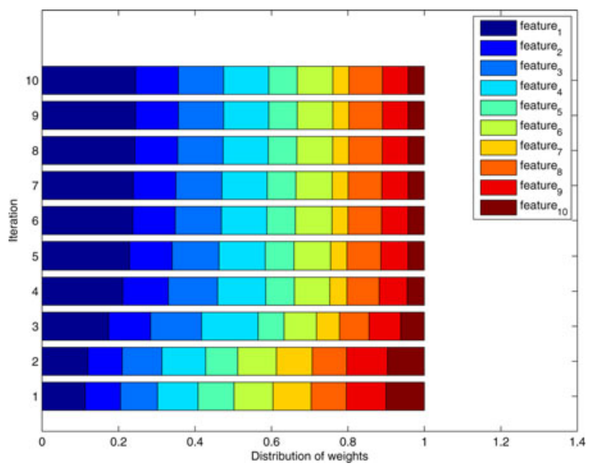


Fig. 3. Evolution of weights for the *Variant* experiment.

all dimensions have roughly equal weights. With each iteration, more weight is added to the more important dimensions (those with lower indexes). Overall, MKFC has a better NMI value (i.e., 0.9192) than Kmean (i.e., 0.9041) and FCM (i.e., 0.9102).

TABLE II
COMPARISONS OF DIFFERENT ALGORITHMS ON UCI DATASETS IN TERMS OF NMI

ID	Kmean	FCM	KFC$_1$	KFC$_2$	KFC$_3$	KFC$_4$	KFC$_5$	KFC$_6$	KFC$_7$	KFC$_8$	MKFC	MKKM
R1	0.865(8)	0.865(8)	0.878(6)	0.893(3)	0.893(3)	0.892(5)	0.877(7)	0.842(10)	0.842(10)	0.811(12)	0.909(1)	0.899(2)
R2	0.320(12)	0.333(11)	0.333(10)	0.335(9)	0.342(6)	0.349(5)	0.356(1)	0.356(2)	0.354(4)	0.335(8)	0.355(3)	0.338(7)
R3	0.139(11)	0.137(12)	0.153(9)	0.201(2)	0.198(3)	0.145(10)	0.167(6)	0.167(6)	0.167(6)	0.178(5)	0.226(1)	0.196(4)
R4	0.570(6)	0.574(3)	0.574(1)	0.573(5)	0.574(4)	0.568(7)	0.563(10)	0.564(9)	0.561(11)	0.556(12)	0.574(2)	0.565(8)
R5	0.125(8)	0.120(10)	0.120(10)	0.115(12)	0.122(9)	0.198(5)	0.197(6)	0.244(1)	0.231(2)	0.211(3)	0.202(4)	0.185(7)
R6	0.581(12)	0.583(11)	0.590(8)	0.596(7)	0.600(6)	0.604(3)	0.608(1)	0.606(2)	0.601(5)	0.586(10)	0.603(4)	0.590(8)
R7	0.577(5)	0.571(7)	0.584(1)	0.584(1)	0.578(3)	0.546(8)	0.527(9)	0.501(10)	0.474(11)	0.409(12)	0.578(3)	0.572(6)
R8	0.121(1)	0.118(6)	0.119(4)	0.119(3)	0.118(5)	0.112(12)	0.114(11)	0.116(7)	0.116(8)	0.116(9)	0.120(2)	0.115(10)
R9	0.980(7)	1.000(1)	1.000(1)	0.992(4)	0.990(5)	0.982(6)	0.975(8)	0.521(10)	0.483(11)	0.325(12)	1.000(1)	0.955(9)
R10	0.806(2)	0.806(2)	0.806(2)	0.806(2)	0.806(2)	0.795(8)	0.784(9)	0.743(10)	0.700(11)	0.607(12)	0.806(2)	0.808(1)
R11	0.803(5)	0.800(7)	0.801(6)	0.804(3)	0.816(1)	0.788(8)	0.755(9)	0.728(10)	0.555(12)	0.573(11)	0.812(2)	0.804(3)
R12	0.664(6)	0.707(1)	0.686(3)	0.683(5)	0.683(4)	0.653(7)	0.616(9)	0.425(10)	0.417(11)	0.403(12)	0.686(2)	0.651(8)
R13	0.102(12)	0.114(10)	0.130(8)	0.130(7)	0.127(9)	0.140(5)	0.144(3)	0.146(1)	0.145(2)	0.143(4)	0.140(6)	0.111(11)
R14	0.362(11)	0.369(9)	0.371(7)	0.371(6)	0.370(8)	0.373(4)	0.372(5)	0.377(3)	0.378(1)	0.234(12)	0.378(2)	0.366(10)
R15	0.253(3)	0.250(10)	0.253(3)	0.254(2)	0.252(8)	0.253(3)	0.253(3)	0.250(9)	0.246(11)	0.238(12)	0.253(3)	0.255(1)
R16	0.454(1)	0.411(2)	0.359(6)	0.361(5)	0.374(3)	0.350(7)	0.347(9)	0.320(11)	0.322(10)	0.302(12)	0.373(4)	0.348(8)
R17	0.722(8)	0.725(7)	0.726(6)	0.729(5)	0.732(2)	0.730(4)	0.722(9)	0.711(10)	0.703(11)	0.700(12)	0.734(1)	0.731(3)
R18	0.707(7)	0.707(7)	0.718(1)	0.718(1)	0.718(1)	0.712(6)	0.702(9)	0.677(10)	0.657(11)	0.635(12)	0.716(5)	0.718(1)
R19	0.387(11)	0.372(12)	0.389(10)	0.401(9)	0.411(8)	0.421(7)	0.441(5)	0.521(1)	0.516(2)	0.509(3)	0.493(4)	0.431(6)
R20	0.374(2)	0.374(1)	0.372(3)	0.371(4)	0.369(5)	0.352(8)	0.338(9)	0.271(10)	0.259(11)	0.243(12)	0.369(5)	0.369(5)
*m*NMI	0.497(7)	0.497(7)	0.498(5)	0.502(3)	0.504(2)	0.498(5)	0.493(9)	0.454(10)	0.436(11)	0.406(12)	0.516(1)	0.500(4)
*m*Rank	6.900(8)	6.850(7)	5.250(4)	4.750(2)	4.750(2)	6.400(6)	6.900(8)	7.100(10)	8.050(11)	9.850(12)	2.850(1)	5.900(5)

Note that in the *Variant* dataset, MKFC performs noticeably better than FCM because MKFC puts more emphasis on important features.

D. University of California, Irvine, Repository

We tested these methods on 20 datasets that are selected from the UCI repository. For each set, only the extracted feature vectors are available—not the raw data. Using the provided feature vectors for data items, we employ different types of kernel functions as bases for multiple kernel learning. These vectors are normalized to have zero mean and unit standard deviation. They are then substituted into the chosen kernels to calculate pairwise distances. As mentioned in Section I, optimal kernel choice is still an open-research topic. Here, following the strategy of other multiple kernel-learning approaches, we select a set of reasonable kernels that are frequently used by kernel methods. In our experiments, we used one polynomial kernel

$$\kappa_k(\mathbf{x}_i, \mathbf{x}_j) = (\theta + \mathbf{x}_i^T \mathbf{x}_j)^p$$

with $\theta = 1$ and $p = 2$, and several Gaussian kernels

$$\kappa_k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)/\sigma).$$

Assume that the minimal value of the Gaussian kernel over the dataset is γ. We then obtain the corresponding σ as

$$\sigma = \min_{i,j}(-(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)/\log(\gamma)).$$

We vary γ over $\{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$ to obtain seven Gaussian kernels. Finally, we normalize the value of each kernel function to the range of $[0.0001. .1]$. We use KFC$_1$, ... , KFC$_8$ to denote KFC's with the aforementioned eight kernels (one polynomial and seven Gaussians), respectively.

We first use hard-clustering measures (i.e., NMI and ARI) to compare Kmean, FCM, KFC, and MKFC. Unlike previous settings for the synthetic data, all the methods (i.e., *k*-means, FCM, KFC, and MKFC) use data with the same dimensions (specified by the #features attribute in Table I). In Table II, we present the average NMI values over 50 runs and the corresponding ranks for different algorithms on the 20 UCI datasets. The numbers in parentheses are the ranks of different methods for each dataset. For example, MKFC ranks 1 with an NMI of 0.909 for the dataset R1, while Kmean ranks 8 with an NMI of 0.865. The last two rows (i.e., *m*NMI and *m*Rank) of Table II display the average NMI value and the average rank for each method over 20 datasets, respectively. MKFC has an average NMI 0.516 and ranks the best of all the methods in terms of average NMI (*m*NMI). In terms of average rank (*m*Rank), MKFC's average rank is 2.85, which is again the best of all the methods. Note that *m*NMI and *m*Rank both yield similar rankings. Table III presents the results of different algorithms in terms of ARI. Changing the measure from NMI to ARI does not change the rankings significantly. Again, MKFC is the best in terms of both *m*ARI and *m*Rank.

It is true that MKFC is not ranked 1 for each individual dataset. However, given the datasets, we do not know in advance which kernel will perform better for each, and there is no single kernel that is suitable for all of them. While kernel combination does not yield the best performance in every single case, on average, it yields the best overall performance. If we were to use a fixed kernel, it would have better performance for some datasets but perform worse in general. This suggests that combining kernels for clustering yields better overall performance than when using a fixed kernel. For real-world applications, we have no cues in advance as to which kernel will work best for the given problem. Despite it is not always ranking the first, MKFC, on average, is the best and yields stable performance.

TABLE III
COMPARISONS OF DIFFERENT ALGORITHMS ON UCI DATASETS IN TERMS OF ARI

ID	Kmean	FCM	KFC$_1$	KFC$_2$	KFC$_3$	KFC$_4$	KFC$_5$	KFC$_6$	KFC$_7$	KFC$_8$	MKFC	MKKM
R1	0.884(7)	0.884(7)	0.899(5)	0.915(3)	0.915(3)	0.897(6)	0.879(9)	0.862(10)	0.862(10)	0.796(12)	0.941(1)	0.931(2)
R2	0.172(10)	0.181(2)	0.182(1)	0.176(6)	0.175(8)	0.179(3)	0.177(5)	0.172(9)	0.161(11)	0.136(12)	0.179(4)	0.176(6)
R3	0.153(11)	0.153(11)	0.239(10)	0.343(2)	0.355(1)	0.306(9)	0.324(4)	0.324(4)	0.324(4)	0.324(7)	0.324(7)	0.342(2)
R4	0.384(2)	0.387(1)	0.379(6)	0.380(4)	0.380(5)	0.371(8)	0.361(9)	0.349(10)	0.342(11)	0.329(12)	0.383(3)	0.379(6)
R5	0.146(1)	0.141(2)	0.133(3)	0.124(4)	0.108(7)	0.098(8)	0.074(9)	0.056(10)	0.033(11)	0.010(12)	0.123(5)	0.109(6)
R6	0.297(12)	0.299(10)	0.307(9)	0.313(7)	0.320(3)	0.321(1)	0.317(4)	0.316(6)	0.317(5)	0.297(11)	0.321(2)	0.312(8)
R7	0.695(7)	0.690(8)	0.701(1)	0.701(1)	0.701(1)	0.695(5)	0.672(9)	0.621(10)	0.573(11)	0.379(12)	0.695(5)	0.701(1)
R8	0.129(11)	0.138(1)	0.136(2)	0.136(2)	0.132(7)	0.135(4)	0.133(6)	0.130(9)	0.127(12)	0.131(8)	0.135(5)	0.130(9)
R9	0.960(4)	1.000(1)	1.000(1)	0.956(5)	0.954(6)	0.798(8)	0.715(9)	0.579(10)	0.472(11)	0.346(12)	1.000(1)	0.954(6)
R10	0.869(1)	0.869(1)	0.869(1)	0.869(1)	0.869(1)	0.859(8)	0.859(8)	0.848(10)	0.838(11)	0.787(12)	0.869(1)	0.869(1)
R11	0.770(9)	0.779(7)	0.785(6)	0.802(3)	0.804(2)	0.794(4)	0.773(8)	0.700(10)	0.674(11)	0.644(12)	0.824(1)	0.794(4)
R12	0.634(6)	0.663(1)	0.652(3)	0.662(2)	0.639(5)	0.619(7)	0.583(9)	0.507(10)	0.480(11)	0.262(12)	0.651(4)	0.610(8)
R13	0.136(2)	0.143(1)	0.135(3)	0.116(4)	0.109(7)	0.099(8)	0.096(9)	0.086(10)	0.081(12)	0.083(11)	0.116(5)	0.112(6)
R14	0.165(12)	0.171(11)	0.173(9)	0.174(5)	0.175(3)	0.174(8)	0.174(7)	0.175(2)	0.175(1)	0.171(10)	0.175(4)	0.174(5)
R15	0.127(9)	0.130(1)	0.130(3)	0.130(4)	0.130(2)	0.129(7)	0.128(8)	0.126(10)	0.122(11)	0.111(12)	0.130(5)	0.130(5)
R16	0.456(1)	0.409(2)	0.333(3)	0.308(6)	0.307(7)	0.320(5)	0.305(8)	0.278(9)	0.245(11)	0.208(12)	0.332(4)	0.265(10)
R17	0.661(1)	0.618(10)	0.639(7)	0.641(6)	0.642(5)	0.650(2)	0.644(4)	0.638(8)	0.614(11)	0.582(12)	0.649(3)	0.637(9)
R18	0.759(6)	0.759(6)	0.761(1)	0.761(1)	0.761(1)	0.755(8)	0.752(9)	0.726(10)	0.693(11)	0.643(12)	0.761(1)	0.761(1)
R19	0.336(8)	0.324(12)	0.334(10)	0.338(7)	0.339(5)	0.345(4)	0.350(2)	0.346(3)	0.356(1)	0.335(9)	0.339(6)	0.330(11)
R20	0.258(1)	0.258(1)	0.258(3)	0.257(4)	0.256(5)	0.250(8)	0.244(9)	0.190(10)	0.169(11)	0.142(12)	0.256(6)	0.255(7)
*m*ARI	0.449(6)	0.450(5)	0.452(4)	0.455(2)	0.454(3)	0.440(8)	0.428(9)	0.402(10)	0.383(11)	0.336(12)	0.461(1)	0.449(6)
*m*Rank	6.050(7)	4.800(5)	4.350(4)	3.850(2)	4.200(3)	6.050(7)	7.250(9)	8.500(10)	9.400(11)	11.200(12)	3.650(1)	5.650(6)

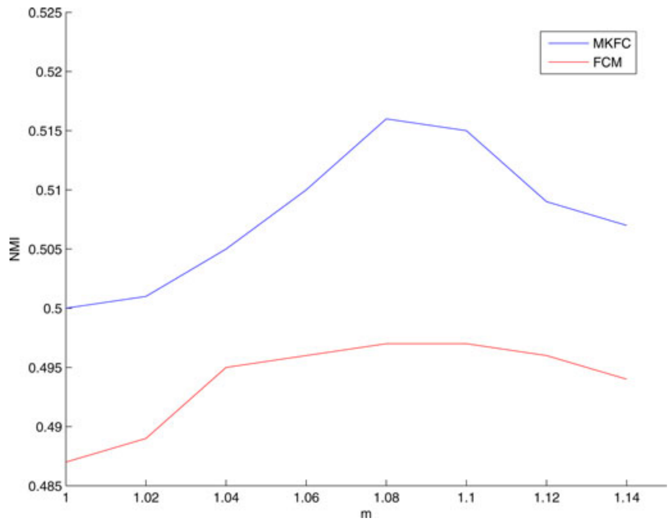


Fig. 4. Average NMI of various m for the UCI experiment.

Like most fuzzy clustering algorithms, choosing the best fuzzification degree m remains an open problem. Graves and Pedrycz [14] had conducted a comparative study on fuzzy clustering and found that different applications and clustering methods may have the best performance with different m's, i.e., the choice of m depends on both applications and clustering algorithms. Fig. 4 shows the performance given various fuzzification degrees m. As mentioned in the experiment setting, we set m to 1.08 in all experiments. Note that both MKFC and FCM perform best around $m = 1.08$, and MKFC is consistently better than FCM for all m's. We can also see that making the clustering a little fuzzy provides better performance than hard clustering.

We also implemented MKKM and compared it with the other methods. MKFC performs better than MKKM in terms of both NMI and ARI. As can be seen, the performance is still improved when there is a slight softness in the clustering. This implies that fuzzy clustering methods may be more able to handle overlapping clusters than the corresponding hard ones, and a slightly larger fuzzification degree may help improve clustering performance.

Again, hard-clustering measures do not necessarily faithfully reflect the performance of soft-clustering algorithms, as they completely ignore membership degrees. As such, we also use the soft-clustering measure EARI to compare FCM, KFC, and MKFC. Kmean and MKKM are omitted, since they are hard-clustering methods. The results in Table IV show that MKFC again ranks first in this measure.

Table V shows the computation time results for averages over 50 runs as the number of iterations and the total time in seconds of each run for the sum of eight KFCs, MKFC, and MKKM on all real datasets, respectively. Both MKFC and MKKM use all eight kernels, while a single KFC only uses one kernel. To ensure a fair comparison, we report the sum of the eight KFCs. For most cases, MKFC requires fewer iterations than the average of the KFCs. This indicates that MKFC converges more quickly. In terms of running time, MKFC took anywhere from less than a second to almost a minute for the test data and was consistently faster than the combined running time of the eight KFCs. MKFC was a bit slower than MKKM, because MKFC uses more iterations to compute the membership of each object, which results in performance gains.

E. Face Clustering

We also evaluated MKFC with face clustering. The face databases are from ORL and CMU-PIE. Fig. 5(a) and (b) shows several sample images of a single person from the ORL and CMU-PIE databases, respectively. The face images are all nearly

TABLE IV
COMPARISON OF DIFFERENT ALGORITHMS ON UCI DATASETS IN TERMS OF EARI

ID	FCM	KFC$_1$	KFC$_2$	KFC$_3$	KFC$_4$	KFC$_5$	KFC$_6$	KFC$_7$	KFC$_8$	MKFC
R1	0.876(6)	0.893(4)	0.902(3)	0.908(2)	0.891(5)	0.875(7)	0.839(8)	0.818(9)	0.751(10)	0.918(1)
R2	0.266(5)	0.266(4)	0.268(2)	0.267(3)	0.265(7)	0.265(6)	0.264(8)	0.264(9)	0.262(10)	0.272(1)
R3	0.149(10)	0.238(9)	0.329(3)	0.353(1)	0.317(8)	0.321(7)	0.322(5)	0.321(6)	0.323(4)	0.350(2)
R4	0.389(1)	0.381(5)	0.382(4)	0.382(3)	0.376(6)	0.366(7)	0.354(8)	0.346(9)	0.332(10)	0.388(2)
R5	0.160(1)	0.155(2)	0.119(6)	0.137(3)	0.120(5)	0.090(7)	0.081(8)	0.052(9)	0.037(10)	0.180(4)
R6	0.327(3)	0.325(9)	0.326(4)	0.323(10)	0.330(2)	0.330(1)	0.326(7)	0.325(8)	0.326(5)	0.327(6)
R7	0.688(1)	0.687(2)	0.686(3)	0.683(4)	0.667(6)	0.655(7)	0.588(8)	0.528(9)	0.357(10)	0.683(5)
R8	0.161(10)	0.165(9)	0.174(8)	0.178(7)	0.191(5)	0.197(4)	0.198(3)	0.200(2)	0.208(1)	0.190(6)
R9	0.992(1)	0.992(1)	0.980(4)	0.977(5)	0.938(7)	0.959(6)	0.795(8)	0.730(9)	0.601(10)	0.992(1)
R10	0.902(1)	0.902(3)	0.899(4)	0.897(5)	0.887(6)	0.881(7)	0.858(8)	0.845(9)	0.802(10)	0.902(1)
R11	0.859(3)	0.868(1)	0.851(4)	0.833(5)	0.827(6)	0.820(7)	0.801(8)	0.792(9)	0.754(10)	0.860(2)
R12	0.637(5)	0.638(4)	0.649(3)	0.652(2)	0.635(6)	0.632(7)	0.616(8)	0.591(9)	0.520(10)	0.661(1)
R13	0.141(1)	0.132(3)	0.114(4)	0.109(5)	0.095(6)	0.087(7)	0.073(8)	0.069(9)	0.068(10)	0.136(2)
R14	0.181(9)	0.181(10)	0.182(7)	0.181(8)	0.183(6)	0.184(3)	0.185(2)	0.184(3)	0.186(1)	0.184(3)
R15	0.129(9)	0.133(1)	0.132(2)	0.130(4)	0.129(10)	0.129(6)	0.130(5)	0.129(8)	0.129(7)	0.132(3)
R16	0.400(1)	0.383(3)	0.337(5)	0.321(6)	0.343(4)	0.314(7)	0.285(8)	0.250(9)	0.215(10)	0.393(2)
R17	0.620(8)	0.641(6)	0.645(5)	0.650(3)	0.661(2)	0.649(4)	0.635(7)	0.618(9)	0.599(10)	0.663(1)
R18	0.722(9)	0.722(9)	0.728(8)	0.730(1)	0.730(1)	0.730(1)	0.728(6)	0.728(6)	0.728(5)	0.730(1)
R19	0.292(10)	0.294(9)	0.315(8)	0.324(7)	0.350(5)	0.355(4)	0.380(3)	0.391(2)	0.400(1)	0.350(6)
R20	0.254(1)	0.254(1)	0.254(3)	0.254(4)	0.254(6)	0.254(7)	0.254(8)	0.254(9)	0.254(10)	0.254(5)
mEARI	0.458(6)	0.463(4)	0.464(3)	0.464(2)	0.460(5)	0.455(7)	0.436(8)	0.422(9)	0.393(10)	0.478(1)
mRank	4.750(4)	4.750(4)	4.500(3)	4.400(2)	5.450(6)	5.600(7)	6.700(8)	7.600(9)	7.700(10)	2.750(1)

TABLE V
NUMBER OF ITERATIONS AND TIME PER RUN FOR THE SUM OF EIGHT KFCs, MKFC, AND MKKM ON ALL REAL DATASETS

ID	KFC iter.	MKFC iter.	MKKM iter.	KFC time	MKFC time	MKKM time
R1	13.6	11.3	12.1	0.066	0.046	0.050
R2	30.8	35.0	28.5	0.267	0.317	0.248
R3	22.6	27.2	20.0	0.127	0.145	0.097
R4	50.4	42.0	38.8	0.797	0.707	0.692
R5	15.2	17.4	14.5	0.127	0.132	0.125
R6	47.6	43.0	41.7	1.361	1.275	1.272
R7	21.0	14.2	11.4	0.479	0.338	0.283
R8	57.5	41.0	35.2	2.058	1.509	1.331
R9	33.8	29.6	23.9	0.329	0.317	0.280
R10	13.8	9.0	9.1	0.127	0.092	0.102
R11	33.8	20.2	16.4	1.737	1.058	0.875
R12	37.8	23.8	22.2	1.973	1.266	1.203
R13	38.2	31.2	25.0	2.012	1.675	1.367
R14	60.4	58.0	49.1	6.542	6.340	5.416
R15	78.3	70.2	61.7	16.219	14.611	12.90
R16	19.1	15.0	12.0	10.785	8.485	6.800
R17	35.7	28.6	22.7	2.423	1.970	1.586
R18	13.5	9.3	8.9	3.834	2.650	2.545
R19	50.5	38.8	33.9	70.611	54.290	47.468
R20	18.2	16.1	13.9	31.238	27.650	23.886

frontal, those in ORL include various facial expressions, and those in CMU PIE include variable lighting conditions. In contrast with the UCI datasets, for this application, we have only the raw data. Thus, our first step is to extract features from the image data. All images were first normalized and cropped to 88×88 pixels. To utilize cues from different perspectives, we extracted three different features.

1) *Eigenface [43]:* After performing principal component analysis, each face image was projected into the eigenspace which preserves 90% of the energy of the eigenvalues.
2) *Gabor texture [44]:* Each face image was filtered with 40 Gabor filters that are generated with five different scales and eight orientations.
3) *LBP [45]:* We used a uniform LBP with eight neighbors and radius 1. Thus, each face image was represented as a 256-bin histogram.

These three features are frequently used for face recognition and represent face images from different perspectives. After extracting these three features, each feature was treated as a vector; these vectors were substituted into the Gaussian kernel to calculate pairwise distances. As with the UCI datasets, we set γ to 0.005. We denote as KFC$_e$, KFC$_g$, and KFC$_l$ the resulting three different kernels from these three features (Eigenface, Gabor texture, and LBP), respectively.

In this experiment, we compared the proposed method (MKFC) with single-KFC and MKKM. Tables VI and VII show the ARI, NMI, EARI, the average number of iterations and the average total time in seconds for ORL and CMU-PIE, respectively. The eigenface, Gabor, and LBP kernel weights as determined by MKFC were 0.175, 0.164, and 0.661 for ORL and 0.626, 0.157, and 0.217 for CMU-PIE. Notably, LBP was more effective for ORL (varying facial expressions), while Eigenface was the best for CMU-PIE (lighting changes). MKFC

(a)

(b)

Fig. 5. Sample images of a subject from (a) ORL and (b) CMU-PIE datasets. Note that ORL exhibits more variation in facial expressions, while CMU-PIE exhibits various lighting conditions.

TABLE VI
COMPARISON OF DIFFERENT METHODS ON FACE DATABASE ORL IN TERMS OF ARI, NMI, EARI, # OF ITERATIONS, AND TIME PER RUN

	KFC$_e$	**KFC**$_g$	**KFC**$_l$	**MKFC**	**MKKM**
ARI	0.046	0.139	0.406	0.464	0.383
NMI	0.369	0.562	0.755	0.783	0.744
eARI	0.108	0.296	0.411	0.462	0.383
Iterations(I)	53.260	5.980	46.180	48.320	36.000
Time(T)	0.625	0.081	0.495	1.7830	1.3284

TABLE VII
COMPARISON OF DIFFERENT METHODS ON FACE DATABASE CMU-PIE IN TERMS OF ARI, NMI, EARI, # OF ITERATIONS, AND TIME PER RUN

	KFC$_e$	**KFC**$_g$	**KFC**$_l$	**MKFC**	**MKKM**
ARI	0.914	0.088	0.597	0.931	0.875
NMI	0.983	0.584	0.845	0.987	0.975
eARI	0.912	0.230	0.587	0.935	0.875
Iterations(I)	24.280	7.300	74.320	25.660	19.020
Time(T)	3.291	0.965	9.955	10.562	7.829

successfully combined the strengths of different features for different datasets and outperformed all other measures for both datasets. As with the UCI datasets, MKKM performed slightly worse than MKFC, showing that fuzzy methods are more able to separate overlapping data than hard methods. The combined computation times for all three KFCs are 1.201 and 14.211 s for ORL and CMU-PIE, respectively, while MKFC took 1.783 and 10.562 s. MKFC was, thus, comparable with the combination of the three KFCs. However, MKFC performs feature selection automatically and provides better clustering results. MKFC was slightly slower than MKKM but yielded better clustering performance.

F. Text Clustering

For text clustering, we used two popular text datasets: 20 Newsgroups and Reuters-21578, which are downloaded from [46]. Each of them is preprocessed by four steps: *all-terms*, *no-short*, *no-stop*, and *stemmed*. We use the datasets *20ng-test-stemmed* and *r8-test-stemmed* to evaluate MKFC. Let $D = \{d_1, \ldots, d_n\}$ be the set of documents and $T = \{t_1, \ldots, t_m\}$ the set of distinct words occurring in D. We denote the frequency of word $t \in T$ in the document $d \in D$ as $tf\,(d,t)$. $tf\text{-}idf$ is a weighting scheme which weights the frequency of a word t in the document d with a factor that discounts its importance with its occurrences in the whole document collection, which is defined as

$$tf\text{-}idf\,(d,t) = tf\,(d,t) \times \log\left(\frac{|D|}{df\,(t)}\right)$$

where $df\,(t)$ is the number of documents in which the word t appears. Thus, the feature vector representation of a document d is defined as

$$\overrightarrow{t_d} = (tf\text{-}idf\,(d,t_1), \ldots, tf\text{-}idf\,(d,t_m)).$$

After normalizing the vectors to a unit length, we used the following four kernels to calculate the pairwise distances between two documents.

1) *Euclidean distance:*

$$\kappa_{ed}(\overrightarrow{t_{d_i}}, \overrightarrow{t_{d_j}}) = \left(\sum_{t=1}^{m} |tf\text{-}idf\,(d_i,t) - tf\text{-}idf\,(d_j,t)|^2\right)^{\frac{1}{2}}.$$

2) *Cosine similarity:*

$$\kappa_{cs}(\overrightarrow{t_{d_i}}, \overrightarrow{t_{d_j}}) = \frac{\overrightarrow{t_{d_i}} \cdot \overrightarrow{t_{d_j}}}{|\overrightarrow{t_{d_i}}||\overrightarrow{t_{d_j}}|}.$$

3) *Jaccard coefficient:*

$$\kappa_{jc}(\overrightarrow{t_{d_i}}, \overrightarrow{t_{d_j}}) = \frac{\overrightarrow{t_{d_i}} \cdot \overrightarrow{t_{d_j}}}{|\overrightarrow{t_{d_i}}|^2 + |\overrightarrow{t_{d_j}}|^2 - \overrightarrow{t_{d_i}} \cdot \overrightarrow{t_{d_j}}}.$$

4) *Pearson correlation coefficient:*

$$\kappa_{pcc}(\overrightarrow{t_{d_i}}, \overrightarrow{t_{d_j}}) = \frac{m \times (\overrightarrow{t_{d_i}} \cdot \overrightarrow{t_{d_j}}) - TF_i \times TF_j}{\sqrt{I \times J}}$$

where

$$TF_i = \sum_{t=1}^{m} tf\text{-}idf\,(d_i,t)$$

$$TF_j = \sum_{t=1}^{m} tf\text{-}idf\,(d_j,t)$$

$$I = m\sum_{t=1}^{m} tf\text{-}idf\,(d_i,t)^2 - TF_i{}^2$$

$$J = m\sum_{t=1}^{m} tf\text{-}idf\,(d_j,t)^2 - TF_j{}^2.$$

Finally, we normalized the value of each kernel function. We denote as KFC$_{ed}$, KFC$_{cs}$, KFC$_{jc}$, and KFC$_{pcc}$ the resulting four kernels, respectively.

Tables VIII and IX show the ARI, NMI, EARI, the average number of iterations and the average total time in seconds

TABLE VIII
COMPARISON OF DIFFERENT METHODS ON TEXT DATASET 20 NEWSGROUPS IN TERMS OF ARI, NMI, EARI, # OF ITERATIONS, AND TIME PER RUN

	KFC$_{ed}$	**KFC$_{cs}$**	**KFC$_{jc}$**	**KFC$_{pcc}$**	**MKFC**	**MKKM**
ARI	0.131	0.140	0.129	0.139	0.143	0.139
NMI	0.381	0.386	0.379	0.391	0.394	0.388
eARI	0.192	0.201	0.187	0.200	0.205	0.139
Iterations(I)	18.100	22.900	19.600	22.880	25.340	19.250
Time(T)	2.559	3.384	2.870	3.371	15.343	11.656

TABLE IX
COMPARISON OF DIFFERENT METHODS ON TEXT DATASET REUTERS-21578 IN TERMS OF ARI, NMI, EARI, # OF ITERATIONS, AND TIME PER RUN

	KFC$_{ed}$	**KFC$_{cs}$**	**KFC$_{jc}$**	**KFC$_{pcc}$**	**MKFC**	**MKKM**
ARI	0.273	0.287	0.236	0.275	0.305	0.275
NMI	0.418	0.428	0.406	0.427	0.434	0.425
eARI	0.311	0.320	0.285	0.303	0.331	0.275
Iterations(I)	5.820	30.440	51.420	30.120	34.440	29.260
Time(T)	0.616	3.135	5.365	3.124	20.331	17.450

for 20 Newsgroups and Reuters-21578, respectively. The kernel weights determined by MKFC were 0.249, 0.250, 0.248, and 0.253 for 20 Newsgroups and 0.248, 0.252, 0.247, and 0.253 for Reuters-21578, respectively. Note that documents are represented with the bag-of-word model, and these four kernels essentially have quite similar clustering capability. Nevertheless, MKFC was still able to assign the weights appropriately to improve the clustering performance.

VI. CONCLUSION

We extended the FCM algorithm to MKFC. The proposed algorithm is easy to implement and provides soft-clustering results that are immune to irrelevant, redundant, ineffective, and unreliable features or kernels. Experiments show that the method effectively incorporates multiple kernels and yields better overall performance. These characteristics make it useful for real-world applications. In the future, we expect to devote our efforts to related open topics, such as strategies for setting the fuzzification degree or choosing the basis kernels.

REFERENCES

[1] R. O. Duda, P. H. Hart, and D. G. Stock, *Pattern Classification*. New York: Wiley, 2001.
[2] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, 2007.
[3] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[4] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 1537–1544.
[5] K. Zhang, I. W. Tsang, and J. T. Kwok, "Maximum margin clustering made practical," in *Proc. 24th Int. Conf. Mach. Learning*, 2007, pp. 1119–1126.
[6] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis*. New York: Wiley, 1999.
[7] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compacy well-separated clusters," *J. Cybern.*, vol. 3, pp. 32–57, 1973.
[8] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Academic, 1981.
[9] H. Shen, J. Yang, S. Wang, and X. Liu, "Attribute weighted mercer kernel based fuzzy clustering algorithm for general non-spherical datasets," *Soft Comput.*, vol. 10, no. 11, pp. 1061–1073, 2006.
[10] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 780–784, May 2002.
[11] D.-Q. Zhang and S.-C. Chen, "Clustering incomplete data using kernel-based fuzzy c-means algorithm," *Neural Process. Lett.*, vol. 18, no. 3, pp. 155–162, 2003.
[12] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[13] B. Zhao, J. Kwok, and C. Zhang, "Multiple kernel clustering," in *Proc. 9th SIAM Int. Conf. Data Mining*, 2009, pp. 638–649.
[14] D. Graves and W. Pedrycz, "Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study," *Fuzzy Sets Syst.*, vol. 161, no. 4, pp. 522–543, 2010.
[15] R. Dave and R. Krishnapuram, "Robust clustering methods: A unified view," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 270–293, May 1997.
[16] R. J. Hathaway and J. C. Bezdek, "Nerf c-means: Non-euclidean relational fuzzy clustering," *Pattern Recognit.*, vol. 27, no. 3, pp. 429–437, 1994.
[17] R. Dave and S. Sen, "Robust fuzzy clustering of relational data," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 713–727, Dec. 2002.
[18] N. Pal and J. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 370–379, Aug. 1995.
[19] J. Yu and M.-S. Yang, "Optimality test for generalized FCM and its application to parameter selection," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 1, pp. 164–176, Feb. 2005.
[20] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, "Low-complexity fuzzy relational clustering algorithms for web mining," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 595–607, Aug. 2001.
[21] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learning Res.*, vol. 2, pp. 299–312, 2001.
[22] F. Camastra and A. Verri, "A novel kernel method for clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 801–804, May 2005.
[23] G. Tzortzis and A. Likas, "The global kernel k-means algorithm for clustering in feature space," *IEEE Trans. Neural Netw.*, vol. 20, no. 7, pp. 1181–1194, Jul. 2009.
[24] J. Leski, "Towards a robust fuzzy clustering," *Fuzzy Sets Syst.*, vol. 137, no. 2, pp. 215–233, 2003.
[25] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 176–190, 2008.
[26] R. Hathaway, J. Huband, and J. Bezdek, "A kernelized non-euclidean relational fuzzy c-means algorithm," in *Proc. 14th IEEE Int. Conf. Fuzzy Syst.*, May, 2005, pp. 414–419.
[27] J.-H. Chiang and P.-Y. Hao, "A new kernel-based fuzzy clustering approach: Support vector clustering with cell growing," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 4, pp. 518–527, Aug. 2003.
[28] G. Lanckriet, T. D. Bie, N. Cristianini, M. Jordan, and W. Noble, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, pp. 2626–2635, 2004.
[29] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learning*, 2004, pp. 6–13.
[30] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf, "Large scale multiple kernel learning," *J. Mach. Learning Res.*, vol. 7, pp. 1531–1565, 2006.
[31] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. 24th Int. Conf. Mach. Learning*, 2007, pp. 775–782.
[32] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proc. 26th Annu. Int. Conf. Mach. Learning*, 2009, pp. 1065–1072.
[33] M. Gonen and E. Alpaydin, "Localized multiple kernel learning," in *Proc. 25th Annu. Int. Conf. Mach. Learning*, 2008, pp. 352–359.
[34] H. Frigui and C. Hwang, "Fuzzy clustering and aggregation of relational data with instance-level constraints," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1565–1581, Dec. 2008.
[35] A. Frank and A. Asuncion. (2010). UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml
[36] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop Appl. Comput. Vis.*, 1994, pp. 138–142.
[37] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1615–1618, Dec. 2003.
[38] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[39] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Mar. 2003.

[40] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?," in *Proc. Int. Conf. Mach. Learning*, Jun. 2009, pp. 1073–1080.

[41] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, ser. KDD, 2009, pp. 877–886.

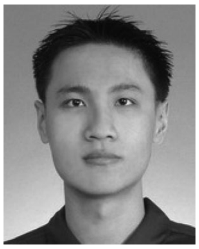
[42] R. J. G. B. Campello, "A fuzzy extension of the rand index and other related indexes for clustering and classification assessment," *Pattern Recognit. Lett.*, vol. 28, no. 7, pp. 833–841, 2007.

[43] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces versus fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[44] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837–842, Aug. 1996.

[45] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[46] A. Cardoso-Cachopo. Datasets for single-label text categorization. (2007). [Online]. Available: http://web.ist.utl.pt/acardoso/



Hsin-Chien Huang received the B.S. and M.S. degrees in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, in 2001 and 2003, respectively. He is currently working toward the Ph.D. degree in computer science and information engineering with National Taiwan University, Taipei, Taiwan.

His research interests include image processing, computer vision, pattern recognition, and multimedia.

Yung-Yu Chuang (M'04) received the B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan, in 1993 and 1995, respectively, and the Ph.D. degree from the University of Washington, Seattle, in 2004, all in computer Science.

He is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include computational photography, content analysis, computer vision, and rendering.

Dr. Chuang is a member of the Association for Computing Machinery.

Chu-Song Chen (S'94–M'96) received the B.S. degree in control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1989 and the M.S. and Ph.D. degrees from the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, in 1991 and 1996, respectively.

He is currently a Deputy Director of the Research Center for Information Technology Innovation, Academia Sinica, Taipei, and a Research Fellow of the Institute of Information Science, Academia Sinica. He is also an Adjunct Professor with the Graduate Institute of Networking and Multimedia, National Taiwan University. His research interests include pattern recognition, computer vision, signal/image processing, and multimedia analysis.

Dr. Chen served as the Secretary General of the Image Processing and Pattern Recognition Society, Taiwan, in which is one of the regional societies of the International Association of Pattern Recognition, during 2007–2008. He has served as the Program Co-Chair of the International Conference on Digital Archive Technologies (ICDAT'2005) and ICDAT'2006, Theme Chair of Pacific-Rim Symposium on Image and Video Technology in 2009, and Area Chair of the Asian Conference on Computer Vision (ACCV'2009 and ACCV'2010), and the Area Chair of the 2010 International Conference on Network-Based Information Systems. He is on the Editorial Board of the *Journal of Multimedia*, *Machine Vision and Applications*, and *IPSJ Transactions on Computer Vision and Applications*.