Adaptive Hypergraph Learning and its Application in Image Classification

Jun Yu, Dacheng Tao, *Senior Member, IEEE*, and Meng Wang, *Member, IEEE*

***Abstract*—Recent years have witnessed a surge of interest in graph-based transductive image classification. Existing simple graph-based transductive learning methods only model the pairwise relationship of images, however, and they are sensitive to the radius parameter used in similarity calculation. Hypergraph learning has been investigated to solve both difficulties. It models the high-order relationship of samples by using a hyperedge to link multiple samples. Nevertheless, the existing hypergraph learning methods face two problems, i.e., how to generate hyperedges and how to handle a large set of hyperedges. This paper proposes an adaptive hypergraph learning method for transductive image classification. In our method, we generate hyperedges by linking images and their nearest neighbors. By varying the size of the neighborhood, we are able to generate a set of hyperedges for each image and its visual neighbors. Our method simultaneously learns the labels of unlabeled images and the weights of hyperedges. In this way, we can automatically modulate the effects of different hyperedges. Thorough empirical studies show the effectiveness of our approach when compared with representative baselines.**

***Index Terms*—Classification, hypergraph, transductive learning.**

I. Introduction

IMAGE classification aims to categorize an image into one of a set of classes. It is one of the most fundamental research topics in image processing, multimedia, and machine learning. A typical process is as follows. First, several labeled training images are collected. Second, classifiers are learned based on the training data. These classifiers are then used to predict labels of new images. Many classification methods have been developed, such as decision tree [1], k-nearest neighbor, and support vector machines (SVMs) [2], [3]. Since image labeling is labor intensive and time consuming, image classification frequently suffers from the training data insufficiency problem. Transductive learning [4], [9], [12], [15]–[17] is a popular approach to address this problem. It explores not only labeled but also unlabeled data, thereby achieving better performance than methods that learn classifiers based only on labeled data.

Among existing transductive learning methods, graph-based learning [5]–[8], [10], [11], [13], [14], [40], [44], [49] achieves promising performance. This type of learning is built on a graph, in which vertices are samples and edge weights indicate the similarity between two samples. A regularization framework is typically formulated by integrating two terms, i.e., a regularizer term that forces the smoothness of the classification results on the graph and a loss term that ensures a low level of training error. These methods intrinsically differ in the graph construction and the definition of the loss term. However, graph-based classification usually encounters two problems.

1) The similarity estimation between samples. The most widely adopted approach is to define the similarity between two samples based on a Gaussian kernel over their distance in the feature space, i.e., $A_{ij} = \exp(-d(x_i, x_j)/\sigma^2)$, wherein σ is a radius parameter. However, existing studies demonstrate that the graph-based learning is usually very sensitive to this parameter; thus, it is not easy to apply graph-based learning in practice.
2) The graph-based learning methods consider only the pairwise relationship between two samples, and they ignore the relationship in a higher order. For example, from a graph, we can easily find two close samples according to the pairwise similarities, but it is not easy to predict whether there are three or more close samples. Essentially, modeling the high-order relationship among samples will significantly improve classification performance.

Hypergraph learning [18] addresses these two problems. Unlike a graph that has an edge between two vertices, a set of vertices is connected by a hyperedge in a hypergraph. Each hyperedge is assigned a weight. In hypergraph learning, the weights of the hyperedges are empirically set according to certain rules. For example, the weight of each hyperedge is simply set to 1 in [18]. In [45], the weight of a hyperedge is calculated by summing up the pairwise affinities within the hyperedge. In practice, we are usually faced with a large number of hyperedges, and these hyperedges have different effects. For example, in the handwriting digit classification [18], a set of hyperedges is generated for each pixel; thus, some hyperedges are redundant. Therefore, weighting or selecting hyperedges will help improve classification performance.

In this paper, we propose an adaptive hypergraph learning method for transductive image classification. For hypergraph construction, we adopt a method that is similar to [25]. Hyperedges are generated based on images and their nearest neighbors. In contrast to the method presented in [25], which adopts

Manuscript received June 11, 2011; revised December 02, 2011 and February 24, 2012; accepted March 01, 2012. Date of publication March 06, 2012; date of current version June 13, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant 61100104 and in part by the Australian Research Council Discovery Project 120103730. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Bulent Sankur.

J. Yu is with the Department of Computer Science, Xiamen University, Xiamen 361005, China.

D. Tao is with the Centre for Quantum Computation and Intelligent Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW 2007, Australia.

M. Wang is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China (e-mail: eric.mengwang@gmail.com).

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/TIP.2012.2190083

a fixed neighborhood size, we vary the size of the neighborhood, whereby multiple hyperedges are generated for each sample. Thus, a large set of hyperedges will be generated in this process. This makes our approach much more robust than previous hypergraph methods, because we do not need to tune the neighborhood size. Since it is difficult to choose a suitable strategy to heuristically weight hyperedges, we consider a principled approach, i.e., regularized loss minimization, based on the statistical learning theory. The loss minimization ensures that the learned weights are optimal for the training set. Since the size of the training set is not large in practice, the regularization item ensures that the learned weights do not overfit to the training samples. This scheme essentially achieves improved classification performance compared with the scheme that uses fixed weights for different hyperedges.

Compared with existing hypergraph learning methods, the proposed method has a regularizer on the hyperedge weights and simultaneously optimizes labels and hyperedge weights. In this way, the effects of different hyperedges can be adaptively modulated. For those hyperedges that are informative, higher weights will be assigned. In addition, the new method sparsifies hyperedge weights, i.e., our method simultaneously weights and selects hyperedges. The contributions of this paper can be summarized below.

1) We propose an adaptive hypergraph learning algorithm for transductive image classification. By simultaneously learning the labels of unlabeled images and optimizing the weights of hyperedges, classification performance can be improved compared with the conventional hypergraph learning method. It is worth emphasizing that the proposed method is a general transductive classification method and can be applied to other tasks. We take image classification for our example.
2) We improve the hypergraph construction approach presented in [25] by varying the size of the neighborhood. This makes our approach much more robust than that in [25], because we do not need to tune the neighborhood size.
3) We conduct comprehensive experiments to empirically analyze our method on six image data sets, including images from different domains. The experimental results validate the effectiveness of our method.

The rest of this paper is organized as follows: Section II briefs transductive learning. Section III reviews conventional hypergraph learning. Section IV details the proposed adaptive hypergraph learning for image classification. Section V shows experiments on six standard data sets. Section VI concludes the paper.

II. Related Work

Transductive learning incorporates not only labeled but also unlabeled samples for improving the classification performance of conventional supervised learning methods. Its success is based on one of the following two assumptions: cluster and manifold assumptions. The cluster assumption supposes that the decision boundary should not cross high-density regions, whereas the manifold assumption means that each class lies on an independent manifold. Many methods have been proposed based on them, e.g., self-training [26], cotraining [27], transductive SVM [28], graph-based learning [29]–[31], and hypergraph learning [18].

Among these methods, hypergraph learning has achieved promising performance in many applications. For example, Agarwal *et al.* [20] applied hypergraph to clustering by using clique average to transform a hypergraph to a simple graph. Zass and Shashua [21] adopted the hypergraph in image matching by using convex optimization. Hypergraph has been applied to problems of multilabel learning [22] and video segmentation [23]. In [24], Tian *et al.* proposed a semisupervised learning method called HyperPrior to classify gene expression data by using biological knowledge as a constraint. In [45], Huang *et al.* formulated the task of image clustering as a problem of hypergraph partition. Each image and its k-nearest neighbors form two kinds of hyperedges based on the descriptors of shape or appearance [45]. In [25], a hypergraph-based image retrieval approach is proposed. This approach constructs a hypergraph by generating a hyperedge from each sample and its neighbors, and hypergraph-based ranking is then performed. Wong and Lu [32] proposed hypergraph-based 3-D object recognition. Bu *et al.* [33] developed music recommendation by modeling the relationship of different entities through a hypergraph to include music, tag, and users. Existing hypergraph learning methods follow the algorithm in [18]. Our paper improves the algorithm in [18] by simultaneously learning the labels of unlabeled samples and the weights of hyperedges, such that image classification performance can be significantly improved.

III. Conventional Hypergraph Learning

This section introduces hypergraph learning theory. In a simple graph, vertices are used to represent samples, and an edge connects two related vertices. The simple graph can be undirected or directed, depending on whether the pairwise relationships among samples are symmetric or not [18]. The learning assignments can be conducted on the graph; for example, we can build an undirected graph based on their pairwise distance, and many graph-based learning methods can be used to classify samples in a feature space. However, simple graphs fail to capture high-order information. Unlike the simple graph, a hyperedge in a hypergraph links several (two or more) vertices. For convenience, Table I lists important notations used in the rest of this paper.

Hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$ is formed by the vertex set $\mathcal{V}$, the hyperedge set $\mathcal{E}$, and the hyperedge weight vector $\mathbf{w}$. Here, each hyperedge e_i is assigned weight $w(e_i)$. A $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix $\mathbf{H}$ denotes $\mathcal{G}$ with the following elements:

$$H(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e. \end{cases} \tag{1}$$

Based on $\mathbf{H}$, the vertex degree of each vertex $v \in \mathcal{V}$ is

$$d(v) = \sum_{e \in E} w(e) H(v,e) \tag{2}$$

and the edge degree of hyperedge $e \in \mathcal{E}$ is

$$\delta(e) = \sum_{v \in V} H(v,e). \tag{3}$$

TABLE I
IMPORTANT NOTATIONS USED IN THIS PAPER AND THEIR DESCRIPTIONS

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$	The representation of a hypergraph, where $\mathcal{V}$ and $\mathcal{E}$ indicate the sets of vertices and hyperedges, respectively, and $\mathbf{w}$ indicates the weight vector of the hyperedges.
$\mathbf{D}_v$	The diagonal matrix of the vertex degrees
$\mathbf{D}_e$	The diagonal matrix of the hyperedge degrees
$\mathbf{H}$	The incidence matrix for the hypergraph
$\mathbf{W}$	The diagonal weight matrix and its (*i*, *i*)-th element is the weight of the *i*-th hyperedge.
$\mathbf{L}$	The constructed hypergraph Laplacian matrix
$\mathbf{Y}_i$	The label vector for *i*-th class. Its *j*-th element is 1 if the *j*-th object belongs to the *i*-th class, and otherwise it is 0.
$\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_c]$	$\mathbf{F}$ represents the relevance score matrix for all samples, and $\mathbf{f}_i$ is the to-be-learned relevance score vector for class *i*.
$\delta(e)$	The degree of the hyperedge *e*
c	The number of classes in the dataset
n	The number of images in the dataset
m	The number of hyperedges
$w(e)$	The weight of hyperedge *e*

We use $\mathbf{D}_v$ and $\mathbf{D}_e$ to denote diagonal matrices of vertex degrees and hyperedge degrees, respectively. Let $\mathbf{W}$ denote the diagonal matrix of the hyperedge weights.

Methods for building the graph Laplacian of hypergraphs can be divided into the following two categories. The first category aims to build a simple graph from the original hypergraph, e.g., clique expansion [46], star expansion [46], and Rodriquez's Laplacian [47]. The second category defines a hypergraph Laplacian by using the analogies from the simple graph Laplacian, e.g., Bolla's Laplacian [48] and normalized Laplacian [18]. It has been verified in [19] that the aforementioned methods are close to each other. In this paper, we adopt the method proposed in [18] to build the hypergraph Laplacian, and the main idea can be directly applied to other methods for hypergraph Laplacian construction.

According to [18], hypergraphs can be applied to different machine learning tasks, e.g., classification, clustering, ranking, and embedding. We take binary classification, which can be formulated as a regularization framework, as an example, i.e.,

$$\arg\min_{f} \{\Omega(f) + \lambda R_{\text{emp}}(f)\} \tag{4}$$

where f is the classification function defined on domain $(-1, 1)$, $\Omega(f)$ is a regularizer on the hypergraph, $R_{\text{emp}}(f)$ is an empirical loss, and $\lambda > 0$ is a tradeoff parameter to balance the empirical loss and the regularizer. In [18], the regularizer on the hypergraph is defined by

$$\Omega(f) = \frac{1}{2} \sum_{e \in E} \sum_{u,v \in V} \frac{w(e)H(u,e)H(v,e)}{\delta(e)} \times \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2. \tag{5}$$

Let $\mathbf{\Theta} = \mathbf{D}_v^{-(1/2)} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-(1/2)}$ and $\mathbf{L} = \mathbf{I} - \mathbf{\Theta}$; the normalized cost function can be rewritten as

$$\Omega(f) = f^T \mathbf{L} f \tag{6}$$

where the hypergraph Laplacian $\mathbf{L}$ is a positive semidefinite matrix.

IV. Adaptive Hypergraph Learning

In this section, we present adaptive hypergraph learning. Fig. 1 shows the workflow of the algorithm. First, features are extracted from images. Labels are available for several images. Second, the hypergraph is constructed on a set of hyperedges generated from each sample and its corresponding neighbors. The hypergraph Laplacian is then used in the transductive classification framework. The labels of unlabeled samples and the weights of the hyperedges are simultaneously optimized through an alternating optimization. Finally, we obtain the labels for classification.

A. Hypergraph Construction via Multiple Neighborhoods

For hypergraph construction, we regard each image in the data set as a vertex on hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$. Assume there are n images in the data set, and thus, the generated hypergraph contains n vertices. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ denote n vertices, and $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ represents m hyperedges. For each hyperedge, there is an associated positive number $w(e)$, which is the weight of the hyperedge e. In our method, a hyperedge is constructed from a centroid vertex and its k nearest neighbors (so the hyperedge connects $k+1$ samples). Instead of generating a hyperedge for each vertex, we generate a group of hyperedges by varying the neighborhood size k in a specified range. Fig. 2 shows four groups of hyperedges with variable neighborhood sizes that are constructed with vertices v_1 to v_4. Taking vertex v_4 for example, three hyperedges $e_1^{v_4}$, $e_2^{v_4}$, and $e_3^{v_4}$ can be constructed with the neighborhood size k setting to 2, 5, and 9, respectively.

With this hypergraph construction approach, a large set of hyperedges are generated with redundancy. In addition, they have different effects in classification. For example, several hyperedges that are generated from samples close to the classification boundary link samples from different classes. Since the regularizer term in hypergraph learning enforces the labels of samples

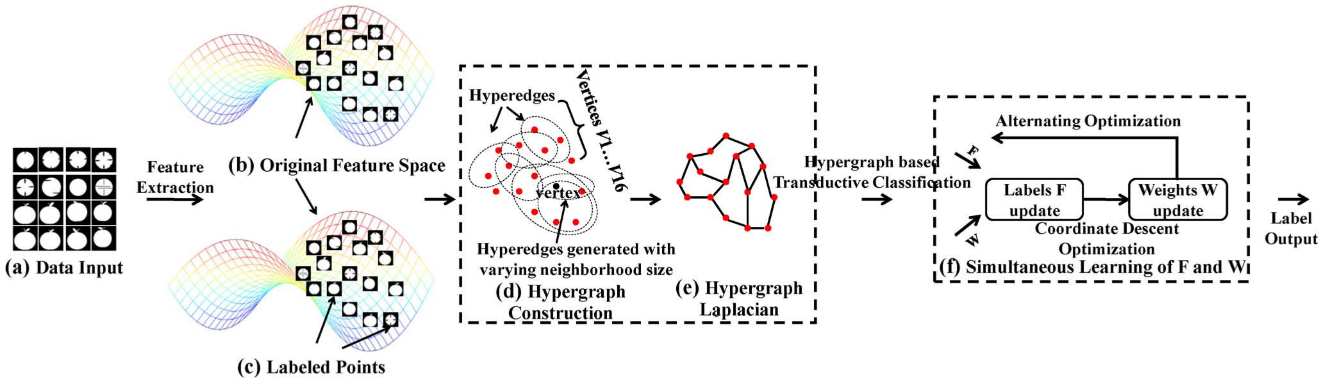

(a) Data Input
Feature Extraction
(b) Original Feature Space
(c) Labeled Points
Hyperedges
Vertices V1…V16
vertex
Hyperedges generated with varying neighborhood size
(d) Hypergraph Construction
(e) Hypergraph Laplacian
Hypergraph based Transductive Classification
Alternating Optimization
Labels F update
Weights W update
Coordinate Descent Optimization
(f) Simultaneous Learning of F and W
Label Output

Fig. 1. Workflow of the adaptive hypergraph learning for image classification. First, features are extracted from all images. Second, the hypergraph is constructed by generating a set of hyperedges from each image and its neighbors. Third, the hypergraph Laplacian is used in the transductive classification framework, and the labels of unlabeled samples and the weights of the hyperedges are simultaneously optimized through an alternating optimization. Finally, image classification is performed based on the obtained labels.

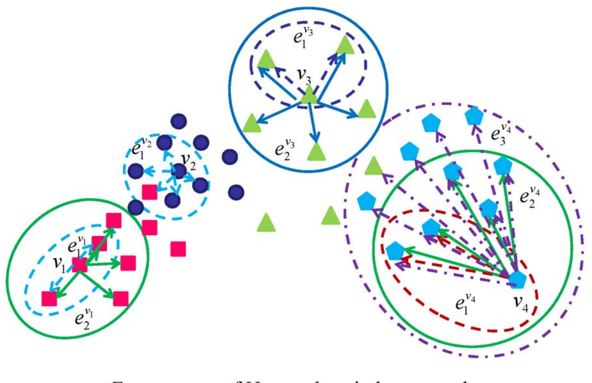


Fig. 2. Schematic illustration of hyperedge generation by varying the neighborhood size. In this example, we show four groups of hyperedges that are constructed for vertices v_1, v_2, v_3, and v_4. For each vertex, we vary the neighborhood size k to generate multiple hyperedges.

connected by a hyperedge to be close, such hyperedges will be less informative or may even have negative effects. In order to automatically modulate the effects of different hyperedges, we introduce a method to learn the labels of unlabeled samples and the weights of the hyperedges simultaneously. We will show that the hyperedge weights are sparse, and thus, our method can select effective hyperedges.

B. Hypergraph Learning for Classification

For a c-class classification problem, according to Section III, we have two terms under the umbrella of the regularization framework, i.e., the regularizer and the classification loss. The regularizer is

$$\Omega(\mathbf{F}) = \sum_{i=1}^{c} f_i^T \mathbf{L} f_i \tag{7}$$

where $\mathbf{f}_i$ is the relevance score vector for the ith class, i.e., the ith column of $\mathbf{F}$. The classification loss is

$$\sum_{i=1}^{c} \|\mathbf{f}_i - \mathbf{y}_i\|^2 = \sum_{i=1}^{c}\sum_{u\in V} (\mathbf{f}_i(u) - \mathbf{y}_i(u))^2 \tag{8}$$

where $\mathbf{y}_i$ is the label vector for the ith class. That means that its jth element is 1 if the jth sample is labeled to be ith class and 0 otherwise. Therefore, the following objective function is defined to obtain $\mathbf{F}$:

$$\arg\min_{\mathbf{F}} \sum_{i=1}^{c} \left(\mathbf{f}_i^T \mathbf{L} \mathbf{f}_i + \lambda \|\mathbf{f}_i - \mathbf{y}_i\|^2\right). \tag{9}$$

C. Adaptive Hypergraph Learning for Classification

Naturally, we can extend the conventional hypergraph learning described in Section III to the adaptive hypergraph learning by simultaneously optimizing $\mathbf{F}$ and $\mathbf{W}$.

According to (5), the weight of hyperedge e is assigned to term $(1/2\delta(e))\sum_{u,v\in V(e)}((f(u)/\sqrt{d(u)}) - (f(v)/\sqrt{d(v)}))^2$ in $\Omega(\mathbf{F})$. Here, $V(e)$ is used to denote the set of vertices connected to e. Therefore, this term measures the label smoothness on the samples in $V(e)$. Since hyperedges connecting to the samples from the same class are informative, a straightforward strategy is to directly minimize (9) with respect to $\mathbf{F}$ and $\mathbf{W}$. In order to control the model complexity motived by the success of sparse learning, we add two constraints $\sum_{j=1}^{m} W_j = 1$ and $W_j \geq 0$. In particular, the first constraint fixes the summation of the weights. The second constraint avoids negative weights. Thus, the solution of $\mathbf{W}$ is on a simplex and enjoys the sparse property, i.e., the weights of useless hyperedges will be set to 0. Therefore, the regularization framework is defined by

$$\arg\min_{\mathbf{F},\mathbf{W}} \sum_{i=1}^{c} \left(\mathbf{f}_i^T \left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{U}\mathbf{W}\mathbf{H}^T\mathbf{D}_v^{-1/2}\right) \times \mathbf{f}_i + \lambda\|\mathbf{f}_i - \mathbf{y}_i\|^2\right)$$
$$\text{s.t.} \sum_{j=1}^{m} W_j = 1, \quad W_j \geq 0, \quad j = 1, \ldots, m \tag{10}$$

where matrix $\mathbf{U}$ is the hyperedge degree $\mathbf{D}_e^{-1}$ with the elements on diagonal entries $(U_1, U_2, \ldots, U_m)$ and diag$(\mathbf{W})$ indicates the diagonal vector of $\mathbf{W}$, i.e., $(W_1, W_2, \ldots, W_m)$.

The aforementioned optimization problem is essentially a linear programming one with respect to $\mathbf{W}$. The solution will be only one weight is 1, and all other weights are 0. To avoid a degenerate solution, we add a two-norm regularizer on the weights. This strategy is popular to avoid overfitting. The regularization framework then becomes

$$\begin{aligned}\arg\min_{\mathbf{F},\mathbf{W}} &\sum_{i=1}^{c}\left(\mathbf{f}_i^T\left(\mathbf{I}-\mathbf{D}_v^{-1/2}\mathbf{HUWH}^T\mathbf{D}_v^{-1/2}\right)\mathbf{f}_i\right)\\ &+\lambda\sum_{i=1}^{c}\|f_i-y_i\|^2+\mu\|\text{diag}(\mathbf{W})\|^2\\ \text{s.t.} &\sum_{j=1}^{m}W_j=1,\quad W_j\geq 0,\quad j=1,\ldots,m. \end{aligned}\tag{11}$$

The regularizer on $\mathbf{W}$ can be also deemed as imposing an assumption that the weights should be equivalent if there is no additional knowledge (the solution of (11) is $W_j = 1/m$ if we remove the first two terms). Although the function is not jointly convex with respect to $\mathbf{F}$ and $\mathbf{W}$, the function is convex with respect to $\mathbf{F}$ with fixed $\mathbf{W}$ or vice versa. Thus, we can use the alternating optimization to efficiently solve the problem.

We first fix $\mathbf{W}$, and we obtain the partial derivative of the objective function in (11) with respect to $\mathbf{F}$, i.e.,

$$\begin{aligned}\frac{\partial}{\partial\mathbf{F}}\left[\sum_{i=1}^{c}\left(\mathbf{f}_i^T\left(\mathbf{I}-\mathbf{D}_v^{-1/2}\mathbf{HUWH}^T\mathbf{D}_v^{-1/2}\right)\mathbf{f}_i\right.\right.\\ \left.\left.+\lambda\|\mathbf{f}_i-\mathbf{y}_i\|^2\right)\right]=0\\ \Rightarrow\mathbf{F}=\frac{\lambda}{1+\lambda}\left(\mathbf{I}-\frac{\mathbf{S}}{1+\lambda}\right)^{-1}\mathbf{Y}\end{aligned}\tag{12}$$

where $\mathbf{S}=\mathbf{D}_v^{-1/2}\mathbf{HUWH}^T\mathbf{D}_v^{-1/2}$. We then fix $\mathbf{F}$ and minimize (11) with respect to $\mathbf{W}$, i.e.,

$$\begin{aligned}&\min_{\mathbf{F}\in\mathcal{R}^{n\times c}}-\sum_{i=1}^{c}\mathbf{f}_i^T\left(\mathbf{D}_v^{-1/2}\mathbf{HUWH}^T\mathbf{D}_v^{-1/2}\right)\mathbf{f}_i+\mu\|\text{diag}(\mathbf{W})\|^2\\ &\text{s.t.}\sum_{j=1}^{m}W_j=1,\quad W_j\geq 0,\quad j=1,\ldots,m.\end{aligned}\tag{13}$$

Since $\mathbf{W}$ and $\mathbf{U}$ are two diagonal matrices, if we denote $\mathbf{F}_i^T\mathbf{D}_v^{-1/2}\mathbf{H}=\mathbf{r}_i=[r_1^i,r_2^i,\ldots,r_m^i]^T$, then the first term $-\sum_{i=1}^{c}\mathbf{f}_i^T(\mathbf{D}_v^{-1/2}\mathbf{HUWH}^T\mathbf{D}_v^{-1/2})\mathbf{f}_i$ in (13) becomes

$$\begin{aligned}&-\sum_{i=1}^{c}\mathbf{r}_i^T*\begin{bmatrix}U_1&0&0\\0&\ddots&0\\0&0&U_m\end{bmatrix}*\begin{bmatrix}W_1&0&0\\0&\ddots&0\\0&0&W_m\end{bmatrix}*\mathbf{r}_i\\ &=-\left(\sum_{i=1}^{c}W_1*\left(r_1^i\right)^2*U_1+\cdots+W_m*\left(r_m^i\right)^2*U_m\right).\end{aligned}\tag{14}$$

Therefore, (13) can be rewritten as

$$\begin{aligned}\min_{\mathbf{W}}\ &W_1*\sum_{i=1}^{c}-\left(r_1^i\right)^2U_1+\cdots\\ &+W_m\sum_{i=1}^{c}-\left(r_m^i\right)^2U_m+\mu\|\text{diag}(\mathbf{W})\|^2\\ \text{s.t.}\ &\sum_{j=1}^{m}W_j=1,\ W_j\geq 0,\ j=1,\ldots,m.\end{aligned}\tag{15}$$

```
Step 1: Initialization
    1.1: Set $\mathbf{W}$ as a diagonal matrix with initial values.
    1.2: Construct the hypergraph Laplacian $\mathbf{L}$ and compute the matrices $\mathbf{D}_v$, $\mathbf{D}_e$ and $\mathbf{H}$ accordingly.
Step 2: Label Update. Compute the optimal $\mathbf{F}$ based on the derivative of (11), which is:
    $\mathbf{F}=\frac{\lambda}{1+\lambda}\left(\mathbf{I}-\frac{\mathbf{S}}{1+\lambda}\right)^{-1}\mathbf{Y}$.
Step 3: Weight Update. Update the weights $\mathbf{W}$ with the iterative coordinate descent method introduced in (16).
Step 4: After obtaining $\mathbf{W}$, update the matrix $\mathbf{S}$ accordingly.
Step 5: Let $t=t+1$. If $t>T$, quit iteration and output the results, otherwise go to step 2.
```

Fig. 3. Implementation of the alternating optimization for obtaining $\mathbf{F}$ and $\mathbf{W}$.

We adopt the coordinate descent algorithm to solve the aforementioned problem. In each iteration, two elements are selected for updating, whereas the others are fixed. For example, in an iteration, the pth and the qth elements, i.e., W_p and W_q, are selected. According to constraint $\sum_{j=1}^{m}W_j=1$, the summation of W_p and W_q will not change after this iteration. Hence, we have

$$\begin{cases}W_p^*=0, W_q^*=W_p+W_q, & \text{if } 2\mu(W_p+W_q)+(s_q-s_p)\leq 0\\ W_p^*=W_p+W_q, W_q^*=0, & \text{if } 2\mu(W_p+W_q)+(s_p-s_q)\leq 0\\ W_p^*=\frac{2\mu(W_p+W_q)+(s_q-s_p)}{4\mu}, & \\ W_q^*=W_p+W_q-W_p^*, & \text{else}\end{cases}\tag{16}$$

where $s_p=\sum_{i=1}^{c}-(r_p^i)^2*U_p$. Note that, in the first line of (16), we can see that W_p^* will be set to 0. This indicates that the weights will be sparse.

Based on the coordinate descent method, we come up with an iterative process that alternately updates the label and the weight. Fig. 3 summaries the implementation of the alternating optimization.

In the aforementioned alternating optimization process, since each step decreases the objective function, which has a lower bound 0, the convergence of the alternating optimization is guaranteed. After obtaining $\mathbf{F}$, the classification of ith sample can be accomplished by assigning it to the kth class that satisfies $k=\arg\min_j F_{ij}$.

D. Discussion

In (11), there are two parameters, i.e., λ and μ, in which λ used by all hypergraph learning algorithms modulates the effect of the loss term and μ is the weighting parameter for the regularizer on the hyperedge weights. If μ is set to 0, the optimal result will be that only one weight is 1 and all other weights are 0, which is a trivial solution. By contrast, if μ tends to be infinity, the solution of (13) will assign identical weights for all hyperedges. Thus, when μ varies between 0 and infinity, the hyperedge weights will vary between an extremely imbalanced case and an extremely balanced case. In our experiments, we use cross validation to choose these two parameters.

Since the alternating optimization approach is not globally optimal, we need to initialize $\mathbf{W}$ with reasonable values. The

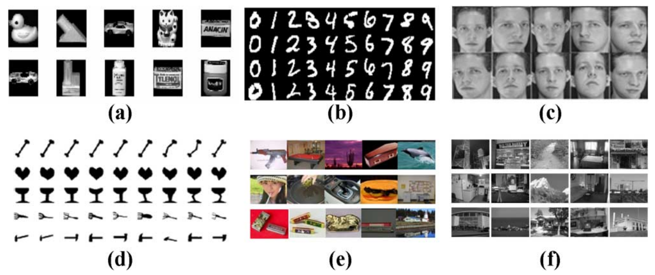


Fig. 4. Sample images from the six image data sets used in the experiment. Sample images from the (a) COIL-20, (b) MNIST, (c) ORL, (d) MPEG-7, (e) Caltech256, and (f) the Scene15 data sets.

TABLE II
DETAILS OF THE SIX DATA SETS USED IN THE EXPERIMENTS

Datasets	Sample size	dimension	Number of classes
COIL-20	1,440	1,024	20
MNIST	2,000	784	10
ORL	400	1,024	40
MPEG-7	700	200	35
Caltech256-2000	2,000	21,504	20
Scene15	1,500	21,504	15

initial value for each hyperedge's weight is set according to the rules used in [25]. First, the $|\mathcal{V}| \times |\mathcal{V}|$ affinity matrix $\mathbf{A}$ is calculated according to

$$A_{ij} = \exp\left(-\frac{\|v_i - v_j\|^2}{\sigma^2}\right) \tag{17}$$

where σ is the average distance among all vertices. Then, the initial weight for each hyperedge is

$$W_i = \sum_{v_j \in e_i} A_{ij}. \tag{18}$$

V. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the proposed approach, we conduct experiments on six image data sets, i.e., COIL-20 [34], handwritten digit image data set MNIST [43], face image data set ORL [36], shape image data set MPEG-7 [38], object image data set Caltech256 [35], and a data set of 15 natural scene categories [37]. Example images from the six image data sets are shown in Fig. 4. Table II summarizes these data sets.

We compare the performance of the proposed adaptive hypergraph learning approach with representative top-level algorithms, such as regular hypergraph learning [25], star-expansion-based hypergraph learning [46], clique-expansion-based hypergraph learning [46], Bolla's Laplacian-based hypergraph learning [48], simple-graph-based learning [29], SVM [2], transductive SVM [28], Laplacian SVM [39], and k-nearest neighbor algorithm. Detailed settings are given below.

A. Data Sets and Configurations

The COIL-20 data set [34] has 1440 images in 20 object categories. Each category contains 72 images captured from different views. All the images are cropped and normalized to 32 × 32 pixel arrays with 256 gray levels per pixel and then scanned to a 1024-D vector.

The MINIST handwritten digit image data set [43] contains two parts, i.e., training and test. We use the test set that contains 2000 images. There are ten classes (digit "0" to "9"), and each image is 28 × 28 in resolution, which results in a 784-D feature vector.

The ORL face image data set [36] contains ten different images for each of the 40 subjects. Each image is described by a 1024-D feature vector. Images are taken at different times, with varied lighting, facial expressions, and facial details.

To evaluate the classification performance on shape image, the 700 images from the MPEG-7 shape image data set [38] are aligned by employing the Hungarian method, and the pairwise distances between images are computed based on the shape context algorithm [41]. Multidimensional scaling is subsequently used to obtain a 200-D feature vector for each shape image.

The Caltech256-2000 is a subset of the Caltech256 data set [35]. It contains 2000 images from 20 categories, i.e., chimp, bear, bowling-ball, light-house, cactus, iguana, toad, cannon, chess-board, lightning, goat, pci-card, dog, raccoon, cormorant, hammock, elk, necktie, and self-propelled-lawn-mower. We adopt the descriptor of locality-constrained linear coding (LLC) [42] to extract features. Each image is described by a 21504-D feature vector.

The Scene15 data set contains 1500 images from 15 natural scenes [37], which are bedroom, CALsuburb, industrial, kitchen, livingroom, MITcoast, MITforest, MIThighway, MITinsidecity, MITmountain, MITopencountry, MITstreet, MITtallbuilding, PARoffice, and store. We use LLC to represent each image.

In experiments, we randomly select several labeled samples. Since the sizes of these data sets widely vary, it is inappropriate to find a fixed number setting for different data sets. Thus, we choose different percentages of samples as labeled data for each data set. For all the classification methods, we independently repeat the experiments five times with randomly selected training samples and report the averaged results with the standard deviation.

B. Comparison Baselines and Configurations

For classification, we compare the following ten methods including the proposed adaptive hypergraph learning.

1) Adaptive hypergraph learning (Ada-HYPER). Parameters λ and μ in (11) are selected by fivefold cross validation. The neighborhood size k varies in {3, 5, 10, 15, 20} in the hyperedge generation process. We set the iteration time in the alternating optimization process to 5.
2) Regular hypergraph learning (HYPER). HYPER [45] does not learn the weights of the hyperedges, which are set according to the method in [25]. Parameter λ in (10) is selected by fivefold cross validation. The neighborhood size k varies in {3, 5, 10, 15, 20}.
3) Star-expansion-based hypergraph learning (Star-HYPER). We adopt the star expansion algorithm in [46] to construct a simple graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ from hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by introducing a new vertex for every hyperedge $e \in \mathcal{E}$. Normalized graph Laplacian is constructed from this new simple graph and is used in transductive classification.

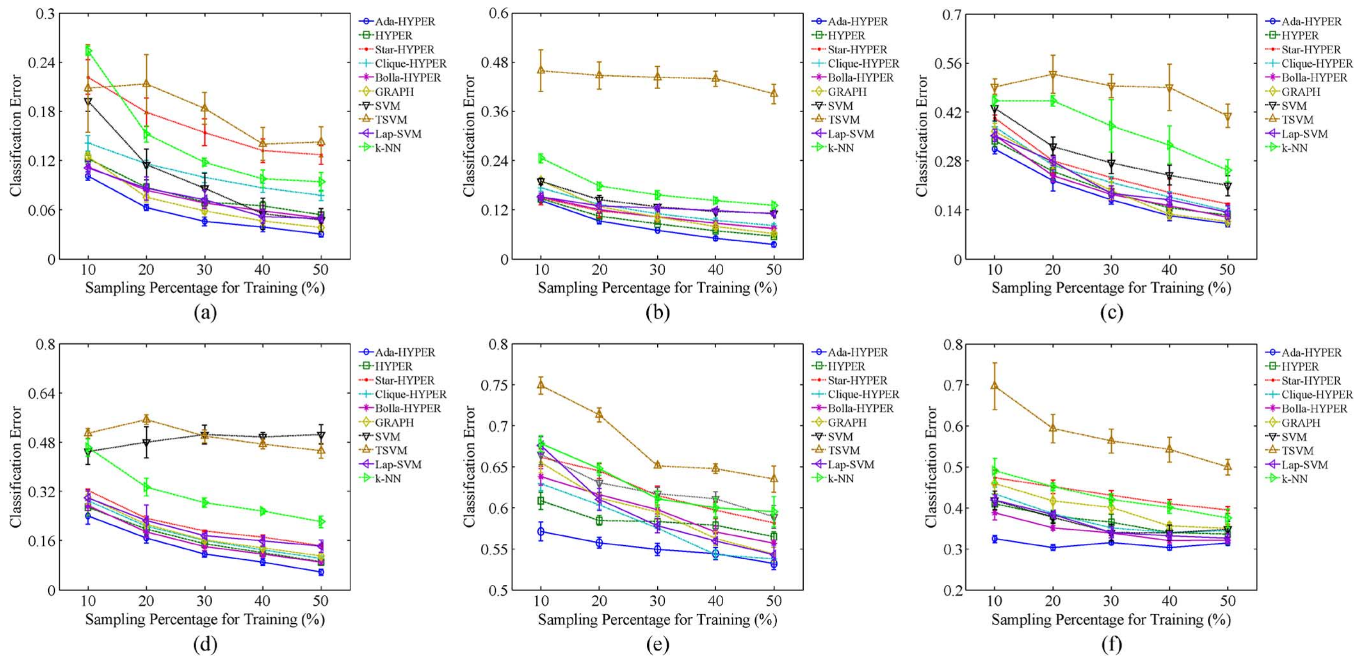


Fig. 5. Classification error rates of different methods (Ada-HYPER, HYPER, GRAPH, SVM, TSVM, Lap-SVM, and k-NN) on the six data sets (in percentage). Results on the (a) COIL-20, (b) MNIST, (c) ORL, (d) MPEG-7, (e) Caltech256, and (f) the Scene15 data sets.

4) Clique-expansion-based hypergraph learning (Clique-HYPER). The clique expansion algorithm [46] constructs a simple graph $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^* \subseteq \mathcal{V}^2)$ from the original hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by replacing each hyperedge with an edge for each pair of vertices in the hyperedge. A normalized graph Laplacian is then constructed from the new simple graph.
5) Bolla's Laplacian-based hypergraph learning (Bolla-HYPER). Bolla's Laplacian [48] is constructed for an unweighted hypergraph in terms of the diagonal vertex degree matrix $\mathbf{D}_v$, the diagonal edge degree matrix $\mathbf{D}_e$, and the incidence matrix $\mathbf{H}$, defined in Section III. The Laplacian matrix is constructed as $\mathbf{D}_v - \mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^T$.
6) Simple-graph-based learning (GRAPH). We adopt the method in [29], which explores normalized graph Laplacian. The radius parameter and the weighting parameter in regularization are tuned to their optimal values.
7) SVM with RBF kernel (SVM) [2]. In order to solve the multiclass problems, we use the one-versus-all strategy. The radius parameter and the weighting parameter in regularization are tuned to their optimal values.
8) Transductive SVM (TSVM) with RBF kernel [28]. In contrast to SVM, TSVM has an additional parameter that modulates the effect of the unlabeled data. All the involved parameters are tuned to their optimal values. We also use the one-versus-all strategy to deal with multiclass problems.
9) Laplacian SVM (Lap-SVM) [39]. This extends the standard SVM by adding an intrinsic smoothness penalty term, which is represented by the graph Laplacian of both labeled and unlabeled data. The involved parameters are tuned to their optimal values.
10) The k-nearest neighbor algorithm (k-NN). This is a method that classifies a sample by finding its closest samples in the training set. In this experiment, parameter k is tuned to the optimal value.

Among the ten methods, SVM and k-NN are inductive classification methods, and Ada-HYPER, HYPER, Star-HYPER, Clique-HYPER, Bolla-HYPER, GRAPH, TSVM, and Lap-SVM are transductive classification methods.

In addition to the aforementioned ten methods, we also compare our approach with the method that generates hyperedges with fixed neighborhood size k. We denote the method as "Ada-HYPER with fixed k." This experiment is conducted to verify the effectiveness of our hyperedge generation method.

C. Experimental Results

Fig. 5 shows the classification results on the six data sets. The proposed Ada-HYPER performs better than HYPER in all cases. Although the superiority is marginal in some cases, it is consistent. This suggests that it is better to simultaneously learn labels and hyperedge weights than to learn only labels. We then compare Ada-HYPER with the other eight methods. Fig. 5 shows that Ada-HYPER achieves the best results in most cases. This demonstrates the effectiveness of the proposed method. We can also see that several methods do not achieve stable performance (for example, although achieving good performance on several data sets, Lap-SVM poorly performs on the Caltech256-2000 data set), but the performance of Ada-HYPER is more stable.

Fig. 6 compares Ada-HYPER with "Ada-HYPER with fixed k." On the Scene15 data set, a larger k tends to be better, but on COIL-20, ORL, and MPEG-7, a smaller k is better. On several other data sets, the performance curves exhibit a "$\bigvee$" shape. This indicates that the optimal k is data dependent. HYPER effectively performs, except for several small k on the ORL and MPEG-7 data sets. In most cases, our method exploring

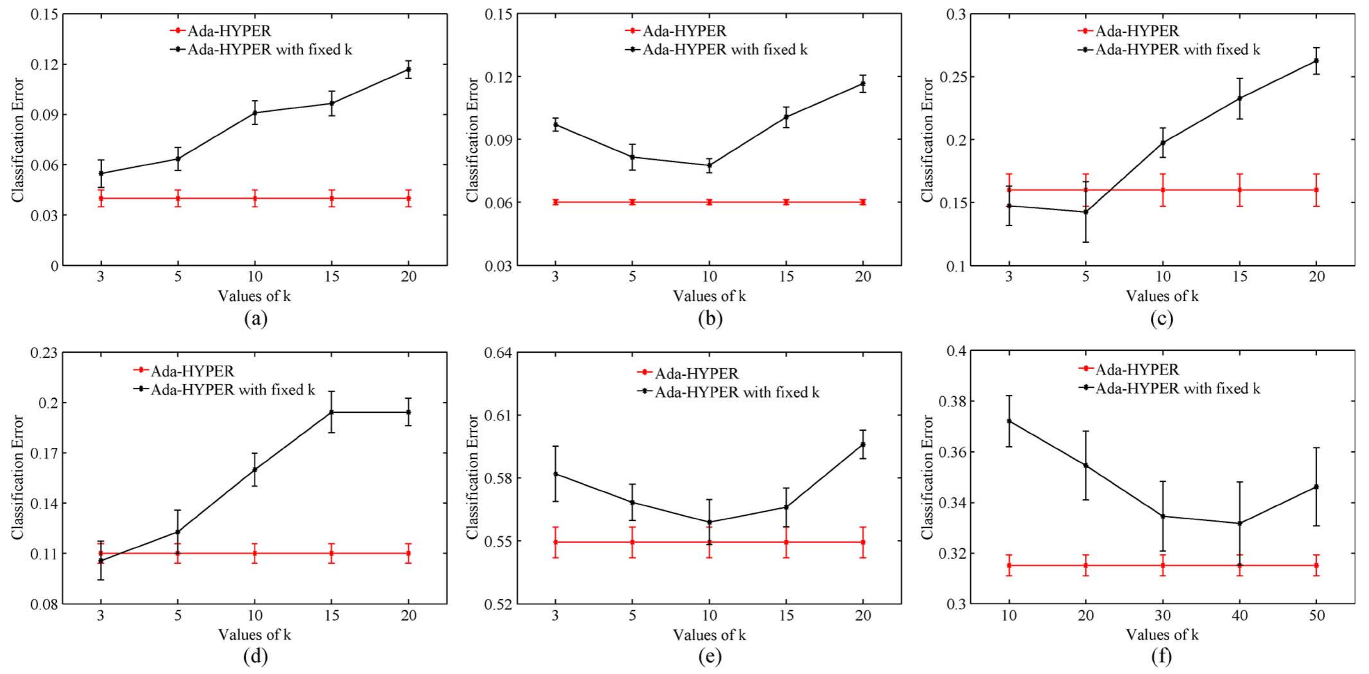


Fig. 6. Performance comparison of Ada-HYPER and Ada-HYPER with fixed k. Here, the sampling percentage for training data is set to 30%. Results on the (a) COIL-20, (b) MNIST, (c) ORL, (d) MPEG-7, (e) Caltech256, and (f) the Scene15 data sets.

TABLE III
NUMBERS OF HYPEREDGES AND THEIR NONZERO WEIGHTS

Datasets	Hyperedge #	Non-Zero Weight #
COIL-20	7,200	106
MNIST	10,000	12
ORL	2,000	39
MPEG-7	3,500	29
Caltech256-2000	10,000	54
Scene15	7,500	57

different neighborhood sizes achieves better performance than using individual best neighborhood size k. This can be also attributed to our hyperedge weight learning approach, because the effects of a rich set of hyperedges can be automatically modulated.

The hypergraph learning method in [45] is closely related to our approach. It selects the region of interest of each image and then constructs hyperedges based on features of shape and appearance. Each vertex (image) and its k-nearest neighbors (based on shape or appearance descriptors) form two kinds of hyperedges. The weight of a hyperedge is computed as the sum of the pairwise affinities within the hyperedge. Its hypergraph learning approach can be deemed as a special case of Ada-HYPER, i.e., with a fixed k and without hyperedge weight learning, Ada-HYPER is reduced to the hypergraph learning proposed in [45]. In our experiments, we have demonstrated the effectiveness of hyperedge weight learning and using multiple neighborhood sizes; thus, the proposed approach essentially improves the method in [45].

Table III shows the numbers of generated hyperedges and the numbers of nonzero weights for the Ada-HYPER method. Only a very small number of hyperedge weights are nonzero. That is because constraints $\sum W_j = 1$ and $W_j \geq 0$ make W_j on a simplex and sparse. Therefore, the proposed Ada-HYPER is able to remove redundant hyperedges while retaining informative ones. Fig. 7 visualizes the values of the hyperedge weights on the six data sets.

We test the sensitivity of parameters λ and μ in Ada-HYPER. In this experiment, we fix the sampling percentage for training data to 30%. Denote by λ_{opt} and μ_{opt} the values obtained by cross validation in our experiments. We first set μ to μ_{opt} and vary λ with $[2^{-4}\lambda_{\text{opt}}, 2^{-3}\lambda_{\text{opt}}, 2^{-2}\lambda_{\text{opt}}, 2^{-1}\lambda_{\text{opt}}, \lambda_{\text{opt}}, 2\lambda_{\text{opt}}, 2^{2}\lambda_{\text{opt}}, 2^{3}\lambda_{\text{opt}}, 2^{4}\lambda_{\text{opt}}]$. The average classification error rates of the 11 data sets are shown in Fig. 8(a). Afterward, we fix λ and vary μ with $[2^{-4}\mu_{\text{opt}}, 2^{-3}\mu_{\text{opt}}, 2^{-2}\mu_{\text{opt}}, 2^{-1}\mu_{\text{opt}}, \mu_{\text{opt}}, 2\mu_{\text{opt}}, 2^{2}\mu_{\text{opt}}, 2^{3}\mu_{\text{opt}}, 2^{4}\mu_{\text{opt}}]$. The average classification error rates are shown in Fig. 8(b). We can see that the average classification error rate varies between 0.208 and 0.231 in Fig. 8(a) and between 0.208 and 0.277 in Fig. 8(b). This experiment demonstrates that the performance of Ada-HYPER will not severely degrade when the parameters vary in a wide range. The performance curves in Fig. 8 have several fluctuations. This is mainly due to the fact that λ_{opt} and μ_{opt}, which are obtained by cross validation, may not be optimal for some data sets. Finally, we analyze the computational cost of Ada-HYPER. We can determine that the computational cost of the hypergraph construction process is $O(dn^2)$. According to Fig. 3, the computational cost of the learning process scales is $O(T_1(n^3 + T_2 m^2))$, where n is the number of images in the data set, m is the number of hyperedges, T_1 is the iteration number in the alternating optimization process, and T_2 is the iteration number in the coordinate gradient descent algorithm.

We compare the computational cost of the Ada-HYPER with those of HYPER [18] and GRAPH [11]. The graph and hypergraph construction process costs for HYPER and GRAPH are $O(dn^2)$ each. For label inference, the cost is $O(n^3)$. Therefore, the cost of these two methods is $O(dn^2 + n^3)$. If T_1 is small (the alternating optimization process usually quickly converges), the

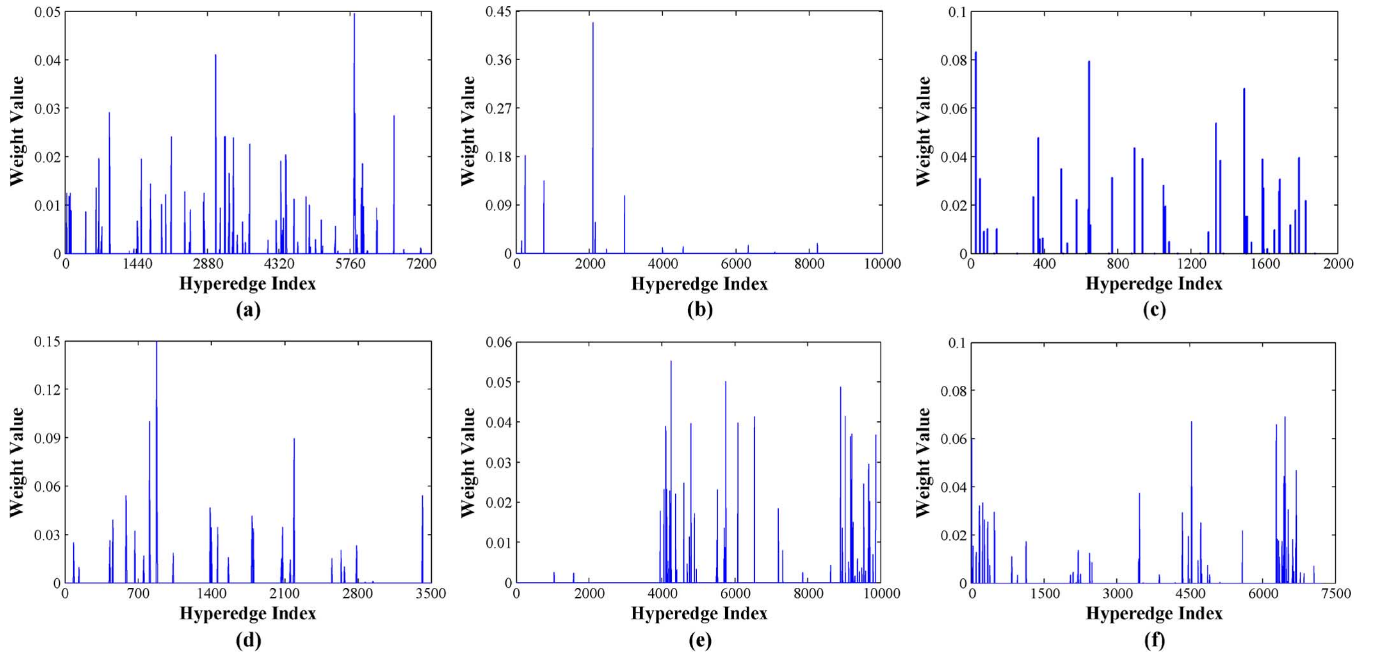


Fig. 7. Illustration of the learned hyperedge weights in the Ada-HYPER method. Illustration for the (a) COIL-20, (b) MNIST, (c) ORL, (d) MPEG-7, (e) Caltech256, and (f) the Scene15 data sets.

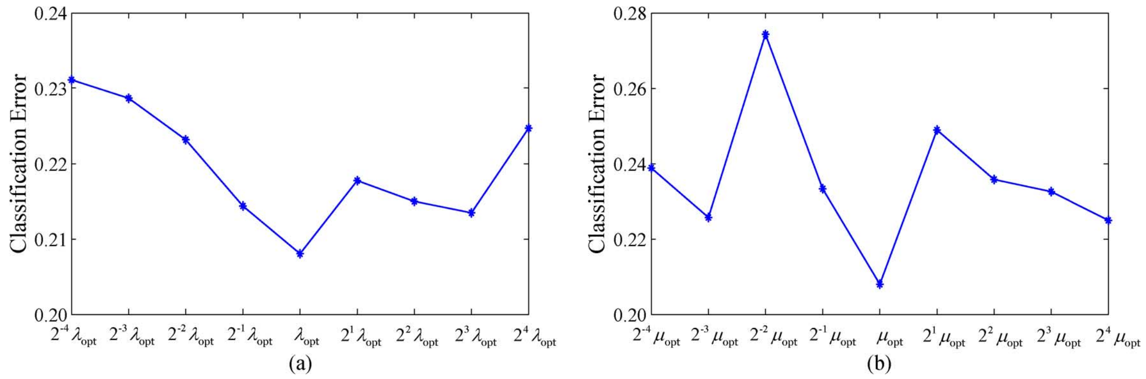


Fig. 8. Average classification error rates with different (a) λ (μ is set to $\mu_{\rm opt}$) and (b) μ (λ is set to $\lambda_{\rm opt}$).

computational cost of Ada-HYPER will be comparable with those of HYPER and GRAPH.

VI. Conclusion

Hypergraph learning is an effective transductive classification framework that avoids two weaknesses in the simple-graph-based transductive learning algorithms, i.e., the failure to model high-order relationships and the tuning of the radius parameter for similarity estimation. In this paper, we have proposed an adaptive hypergraph learning approach for transductive image classification. The approach not only investigates a robust hyperedge construction method but also presents a simultaneous learning of the labels of unlabeled images and the weights of hyperedges. In the hypergraph construction, we have generated a hyperedge to link an image with its varying-size neighborhood. In the learning process, an alternating optimization method is introduced to optimize both the labels and hyperedge weights.

We have conducted experiments on six image data sets. We compare the proposed adaptive hypergraph learning with representative inductive and transductive classification algorithms. Experimental results demonstrate that the proposed method outperforms other methods in most cases. In particular, it consistently performs better than the conventional hypergraph learning algorithm, and this result suggests the superiority of the simultaneous learning of labels and hyperedge weights. In addition, we have empirically shown that generating hyperedges with different neighborhood sizes can be better than using only a fixed neighborhood size.

The analysis in Section V-C shows that the computational cost of the adaptive hypergraph learning approach cubically scales with respect to the number of samples and is comparable with those of the conventional graph and hypergraph learning algorithms. This leads to the difficulty of the algorithm in handling large-scale data sets. However, several strategies have been developed for graph and hypergraph learning algorithm for reducing the computational cost, such as sparsifying $\mathbf{L}$, adopting an iterative process to solve (7), and exploring a set of anchor points [14] to reduce $\mathbf{L}$. Essentially, these strategies are readily applicable to Ada-HYPER for handling large-scale databases. In the future, we will empirically study these approaches and develop new strategies to make the adaptive hypergraph learning efficient for large-scale databases.

REFERENCES

[1] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 4, pp. 476–487, Nov. 2005.

[2] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ: Wiley-Interscience, 1998.

[3] F. Bovolo, L. Bruzzone, and L. Carlin, "A novel technique for subpixel image classification based on support vector machine," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2983–2999, Nov. 2010.

[4] J. Yu, D. Liu, D. Tao, and S.-H. Soon, "Complex object correspondence construction in 2D animation," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3257–3269, Nov. 2011.

[5] D. Song and D. Tao, "Biologically inspired feature manifold for scene classification," *IEEE Trans. Image Process.*, vol. 19, no. 1, pp. 174–184, Jan. 2010.

[6] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Manifold regularized discriminative non-negative matrix factorization with fast gradient descent," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 2030–2048, Jul. 2011.

[7] X. Zhu, Z. Ghaharmani, and J. Lafferty, "Semisupervised learning using Gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.

[8] M. Belkin, L. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Proc. Conf. Comput. Learn. Theory*, Banff, Canada, 2004, pp. 624–638.

[9] O. Chapelle, A. Zien, and B. Scholkopf, *Semi- Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[10] H. Shin, J. Hill, and G. Ratsch, "Graph- based semi-supervised learning with sharper edges," in *Proc. Eur. Conf. Mach. Learn.*, Berlin, Germany, 2006, pp. 401–412.

[11] X. Zhu, "Semi-supervised learning with graphs," M.S. thesis, Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, 2005, CMU-LTI-05-192.

[12] X. Zhu, "Semi-supervised learning literature survey," Univ. Wisconsin-Madison, Madison, Tech. Rep., 2005.

[13] M. Culp and G. Michailidis, "Graph based semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 174–179, Jan. 2008.

[14] W. Liu, J. He, and S. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 679–686.

[15] G. Druck and A. McCallum, "High-performance semi-supervised learning using discriminatively constrained generative models," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 319–326.

[16] R. He and W. Zheng, "Nonnegative sparse coding for discriminative semi-supervised learning," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, Colorado Springs, CO, 2011, pp. 2849–2856.

[17] P. Mallapragada, R. Jin, A. Jain, and Y. Liu, "SemiBoost: Boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, Nov. 2009.

[18] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2006, pp. 1601–1608.

[19] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in *Proc. Int. Conf. Mach. Learn.*, Pittsburgh, PA, 2006, pp. 17–24.

[20] S. Agarwal, J. Lim, L. Zelnik Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, San Diego, CA, 2005, pp. 838–845.

[21] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, Anchorage, AK, 2008, pp. 1–8.

[22] L. Sun, S. Ji, and J. Ye, "Hypergraph spectral learning for multi-label classification," in *Proc. Int. Conf. Know. Discov. Data Mining*, Las Vegas, NV, 2008, pp. 668–676.

[23] Y. Huang, Q. Liu, and D. Metaxas, "Video object segmentation by hypergraph cut," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, Miami, FL, 2009, pp. 1738–1745.

[24] Z. Tian, T. Hwang, and R. Kuang, "A hypergraph-based learning algorithm for classifying gene expression and array CGH data with prior knowledge," *Bioinformatics*, vol. 25, no. 21, pp. 2831–2838, Jul. 2009.

[25] Y. Huang, Q. Liu, S. Zhang, and D. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, San Francisco, CA, 2010, pp. 3376–3383.

[26] C. Rosenberg, M. Heberg, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. Workshop Appl. Comput. Vis.*, Breckenridge, CO, 2005, pp. 29–36.

[27] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. Workshop Comput. Learn. Theory*, Madison, WI, 1998, pp. 92–100.

[28] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. Int. Conf. Mach. Learn.*, Bled, Slovenia, 1999, pp. 200–209.

[29] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2004, pp. 321–328.

[30] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[31] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[32] A. Wong and S. Lu, "Recognition and shape synthesis of 3D objects based on attributed hypergraphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 3, pp. 279–290, Mar. 1989.

[33] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, "Music recommendation by unified hypergraph: Combining social media information and music content," in *Proc. ACM Int. Conf. Multimedia*, Firenze, Italy, 2010, pp. 391–400.

[34] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (COIL-20)," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CUCS-005-96, Feb. 1996.

[35] G. Griffin, A. Holub, and P. Perona, Caltech-256 California Inst. Technol., Pasadena, CA, Caltech Technical Report, 2007.

[36] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. Workshop Appl. Comput. Vis.*, Sarasota, FL, 1994, pp. 138–142.

[37] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. Comput. Vis. Pattern Recog.*, New York, 2006, pp. 2169–2178.

[38] "Shape data for the MPEG-7 core experiment CE-Shape-1," [Online]. Available: http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz

[39] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006.

[40] M. Wang, X. Hua, R. Hong, J. Tang, G. Qi, and Y. Song, "Unified video annotation via multi-graph learning," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.

[41] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[42] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. Comput. Vis. Pattern Recog.*, San Francisco, CA, 2010, pp. 3360–3367.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[44] M. Wang, X. Hua, J. Tang, and R. Hong, "Beyond distance measurement: Constructing neighborhood similarity for video annotation," *IEEE Trans. Multimedia*, vol. 11, no. 3, pp. 465–476, 2009.

[45] Y. Huang, Q. Liu, F. Lv, Y. Gong, and D. Metaxas, "Unsupervised image categorization by hypergraph partition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1266–1273, Jun. 2011.

[46] J. Y. Zien, M. D. F. Schlag, and P. K. Chan, "Multi- level spectral hypergraph partitioning with arbitrary vertex sizes," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 1996, pp. 201–204.

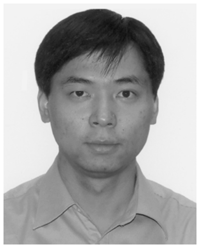
[47] J. Rodrequez, "On the Laplacian spectrum and walk-regular hypergraphs," *Linear Multilinear Algebra*, vol. 51, no. 3, pp. 285–297, 2003.

[48] M. Bolla, "Spectra, Euclidean representations and clusterings of hypergraphs," *Discrete Math.*, vol. 117, no. 1–3, pp. 19–39, Jul. 1993.

[49] M. Wang, K. Yang, X.-S. Hua, and H.-J. Zhang, "Towards a relevant and diverse search of social images," *IEEE Trans. Multimedia*, vol. 12, no. 8, pp. 829–842, 2010.

Jun Yu received the B.E. and Ph.D. degrees from Zhejiang University, Hangzhou, China, respectively.

He is an Associate Professor with Xiamen University, Xiamen, China. His current research interests include computer graphics, computer vision, machine learning, and multimedia. He has authored and coauthored more than 20 journal and conference papers in these areas.



Dacheng Tao (M'07–SM'12) received the B.Eng. degree from the University of Science and Technology of China, Hefei, China, the M.Phil. degree from the Chinese University of Hong Kong, Shatin, Hong Kong, and the Ph.D. degree from the University of London, London, U.K.

He is Professor of computer science with the Centre for Quantum Computation and Information Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. He has authored and coauthored more than 100 scientific articles at top venues including IEEE TPAMI, TIP, AISTATS, CVPR, and ICDM. He mainly applies statistics and mathematics for data analysis problems in computer vision, machine learning, multimedia, data mining, and video surveillance.

Dr. Tao was a recipient of the Best Theory/Algorithm Paper Runner-Up Award in the IEEE ICDM'07.

Meng Wang (M'09) received the B.E. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China.

He was an Associate Researcher with Microsoft Research Asia, and then a Core Member in a startup in Silicon Valley. After that, he was a Senior Research Fellow with the National University of Singapore. He is a Professor with Hefei University of Technology, Hefei. His current research interests include multimedia content analysis, search, mining, recommendation, and large-scale computing. He has authored more than 100 book chapters, journal, and conference papers in these areas.

Dr. Wang was a recipient the Best Paper Awards continuously in the 17th and 18th Association for Computing Machinery (ACM) International Conference on Multimedia and the Best Paper Award in the 16th International Multimedia Modeling Conference. He is a member of the ACM.