MTC: A Fast and Robust Graph-Based Transductive Learning Method

Yan-Ming Zhang, Kaizhu Huang, *Member, IEEE*, Guang-Gang Geng, and Cheng-Lin Liu, *Senior Member, IEEE*

***Abstract*— Despite the great success of graph-based transductive learning methods, most of them have serious problems in scalability and robustness. In this paper, we propose an efficient and robust graph-based transductive classification method, called minimum tree cut (MTC), which is suitable for large-scale data. Motivated from the sparse representation of graph, we approximate a graph by a spanning tree. Exploiting the simple structure, we develop a linear-time algorithm to label the tree such that the cut size of the tree is minimized. This significantly improves graph-based methods, which typically have a polynomial time complexity. Moreover, we theoretically and empirically show that the performance of MTC is robust to the graph construction, overcoming another big problem of traditional graph-based methods. Extensive experiments on public data sets and applications on web-spam detection and interactive image segmentation demonstrate our method's advantages in aspect of accuracy, speed, and robustness.**

***Index Terms*— Graph-based method, large-scale manifold learning, semisupervised learning (SSL), transductive learning (TL).**

I. Introduction

THE purpose of semisupervised learning (SSL) is to improve the accuracy of supervised learning by exploiting both labeled and unlabeled data. Over the past decades, a large family of SSL methods have been proposed. The key idea underlying these methods is to exploit the structure of input space to improve the prediction of data labels. The earliest approaches adopted the generative approach [31], [33], [38]. They assume that the data (x, y) are generated from a mixture model, and there is a one-to-one correspondence between the mixture components and classes. Co-training [6] is another important SSL method that assumes that the data contains multiple conditionally independent and sufficient views (feature sets). When the condition is satisfied, co-training can lead to elegant theoretical and empirical results [6], [16], [32], [37]. Many of its variants were developed [32], [39], [49].

Manuscript received December 20, 2013; revised July 30, 2014; accepted October 6, 2014. Date of publication November 4, 2014; date of current version August 17, 2015. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2012CB316300 and in part by the National Natural Science Foundation of China under Grant 61203296, Grant 61473236, and Grant 61375039.

Y.-M. Zhang and C.-L. Liu are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: ymzhang@nlpr.ia.ac.cn; liucl@nlpr.ia.ac.cn).

K. Huang is with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China (e-mail: kaizhu.huang@xjtlu.edu.cn).

G.-G. Geng is with the China Internet Network Information Center, Chinese Academy of Sciences, Beijing 100190, China (e-mail: gengguanggang@cnnic.cn).

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/TNNLS.2014.2363679

More recently, researchers made more sophisticated use of unlabeled data based on the cluster assumption and manifold assumption. The cluster assumption states that data from the same class form high-density clusters in the input space; therefore, the decision boundary should only lie in low-density regions. Inspired by this, many SSL approaches [12], [19], [23], [27] have been proposed. Despite this clear motivation, these methods, such as transductive support vector machines (TSVMs) [23], always result in complex objective functions, which make the training phase very difficult. Actually, a majority of works on TSVM are on designing efficient algorithms to solve the optimization problem. See [11] for a detailed review.

On the other hand, the manifold assumption states that data points located on the same low-dimensional manifold should share the same label. Since this class of methods always construct a graph as the discrete approximation of the data manifold, they are also called as graph-based SSL methods [2]–[4], [24], [48], [51]. Graph-based methods are among the most popular SSL methods. On one hand, graph provides a powerful tool to describe the input structure of data and methods based on graphs always lead to promising results. On the other hand, in many applications, like social network, genomic data, web pages, and so on, the data are presented directly as graphs without explicitly given feature vectors; thus, graph-based methods are important on their own right.

Despite these advantages, graph-based methods suffer from two main drawbacks [22], [30], [50].

1) They are not scalable. Most graph-based methods have a time complexity of $O(n^3)$ because of the calculation of the inverse of graph Laplacian [48], [51] or the spectral decomposition of graph Laplacian [2], [24]. Although the complexity can be significantly reduced when the graph is sparse, it is still polynomial in the number of edges [40], which is computationally formidable to large-scale problems.
2) Their performance is very sensitive to the graph construction. For the most popular k-nearest-neighbor (kNN) graph or ϵ graph, a small change in k or ϵ would make a big difference in accuracy [14], [15], [22], [43]. For this reason, one has to carefully tune these parameters, which is, however, impractical because of the scarcity of labeled data.

In this paper, we consider a particular setting of SSL called transductive learning (TL), and propose a fast and robust graph-based TL method that overcomes the aforementioned problems. Recall that, SSL and TL are different in that the

output of a SSL method is a decision rule that can be used for classification of out-of-sample data, while the output of a TL method is prediction results of the unlabeled points in the in-sample data set. Specifically, we are given l labeled data points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ and u unlabeled data points $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$, where $\mathbf{x}_i \in \mathbb{R}^d$ $(1 \leq i \leq l+u)$ is the input of a data point, $y_i \in \{1, \ldots, K\}$ $(1 \leq i \leq l)$ indicates the class label of x_i, and K is the number of classes. The goal is to predict the labels of unlabeled data.

Our work is motivated by the following observation: most graphs contain large number of redundant edges that can be safely removed without scarifying the accuracy of the TL method. With the idea of graph sparsification, our method first approximates the graph with a spanning tree, and then labels the tree in a way such that the overall cut size is minimized. Compared with existing methods, the contributions of this paper lie in three aspects.

1) For a given spanning tree, our labeling algorithm has a linear time and space complexity with respect to the data size, thus is suitable for large-scale problems. For example, on a graph with 4 096 000 nodes and 163 465 800 edges, the running time of our method is about 71 s on a PC (Section VI-B). Furthermore, extensive empirical results show that the accuracy of our method approximates the full graph-based methods, while is better than other scalable methods.
2) We theoretically and empirically show the robustness of our method. We find the structure of a minimum spanning tree (MST) is robust to the graph construction. For a connected ϵ graph with the radial basis function (RBF) weighting function, we formally prove that the structure of MST is invariant to the variance of ϵ and σ. We also observe the same property empirically for the kNN graph. This property in turn makes the performance of our method very robust to the graph hyperparameter. To the best of our knowledge, this is the first work to extensively explore the robustness property of tree-based SSL methods.
3) Besides its robustness and speed, our method has no hyperparameter, such as regularization factor, learning rate, or stopping criterion, to tune. These make it an off-the-shelf technique for practical problems.

Note that, although there are works exploiting a tree to approximate a graph [9], [21], [42], the tree-based learning model and labeling algorithm proposed in this paper are new; this leads to a consistent improvement of our approach to these methods. Please refer to Sections II-A and V for more details.

This paper is an extension of [47] by reorganizing the method description with additional theorems, and particularly, redesigning experiments and applications to evaluate on extensive data sets in comparison with more existing methods.

The rest of this paper is organized as follows. Section II reviews the related work. Section III presents our major work, including the model definition and the detailed method. After that, Section IV gives the robust analysis of the method. Section V presents comparison experimental results to validate the advantages of our method. In Section VI, we apply the method to two real-world problems: web-spam detection and interactive image segmentation. Finally, we set out the conclusion in Section VII.

II. Related Work

In this section, we give a brief introduction to the existing works devoted to the scalable and robust SSL. A more comprehensive review can be found in [29].

A. Scalable Graph-Based SSL/TL Methods

A lot of works have been done to reduce the computational complexity of original graph-based SSL algorithms. The central idea is to simplify the structure of the graph, and the methods can be divided into two categories.

The first class of scalable graph-based SSL methods is based on subsampling a small subset of data points [41]. These prototypes are used to build both the prediction function and the adjacency matrix. Methods differ from each other in the approaches to construct the adjacency matrix. Delalleau *et al.* [17] built the matrix by directly setting $w_{ij} = 0$ if neither i nor j is a prototype. Zhang *et al.* [46] approximated the adjacency matrix by the Nyström method, and Liu *et al.* [28] approximated it by a two-step transition probability matrix. The running time of all these prototype-based methods is $O(m^2 n)$, where m is the number of prototypes. However, these methods need to use feature vectors of data and are hence incapable of handling problems where only graphs are available. Moreover, the selection of prototypes is not trivial.

Recently, another class of scalable methods has been proposed, which is based on subsampling a small subset of edges of the graph. Herbster *et al.* [20] considered the online graph prediction problem. They proposed to approximate a general graph with a line graph, and use the nearest neighbor classifier to make prediction on it. Later, Cesa-Bianchi *et al.* [10] extended [20] from the unweighted graph to the weighted one. The cost of both algorithms is $O(n)$ in both time and space requirement. However, approximating graphs by lines may lose too much information, and degrade the prediction accuracy. Experimental results later presented in Section V also validate this point.

Herbster *et al.* [21] proposed to approximate a graph using a spanning tree, and designed an algorithm to calculate the inverse of the tree's Laplacian in $O(n^2)$ time. Then, the perceptron algorithm is performed to train a linear classifier in the reproducing kernel Hilbert space space induced by this Laplacian kernel. By calculating one column of the Laplacian kernel for each trial, their algorithm takes $O(ln)$ time and $O(n)$ space. Cesa-Bianchi *et al.* [9] also adopted the idea of approximating graphs by trees, but proposed to label the tree using a mixed strategy: the unlabeled fork nodes were predicted by minimizing the cut size of tree, while other unlabeled nodes were predicted by the nearest neighbor rule. This method can only be applied to unweighted graph. Vitale *et al.* [42] extended it to handle the weighted graphs. Both the time and space cost of the two algorithms are $O(n)$.

B. Scalable Graph Construction Methods

The running time for building the kNN graph or ϵ-graph is $O(n^2)$, which is impractical for large n. Thus, efficient graph building algorithms is another key element for developing scalable graph-based SSL methods. Chen *et al.* [13] and Wang *et al.* [44] proposed methods for building approximate kNN graphs by divide-and-conquer strategy using splitting trees. Dong *et al.* [18] proposed another approximate kNN graph construction method based on the local search and neighbor propagation strategies. Liu *et al.* [28] first select a set of representative data points as anchor points, then build the graph by connecting each data to its nearest anchor points. Saluja and Kveton [36] directly construct cover trees from vector data, then apply classic SSL methods on the resulting trees.

C. Robust Graph-Based SSL/TL Methods

Although it is well known that existing graph-based SSL methods are very sensitive to the graph building step, work to enhance the robustness of graph-based SSL methods is relatively rare. The common idea is to use better criterions to construct graph so as to overcome the weakness of kNN graph and ϵ graph.

Wang and Zhang [43] proposed to learn the weights for each node through reconstruction error minimization using its predefined neighborhood. Cheng *et al.* [14] exploited the similar idea, but the point's neighborhood was automatically learned using l_1 norm. Daitch *et al.* [15] also minimized the data's reconstruction error to learn the weights but formulated one optimization problem for the whole data set. In kNN graph, the number of a node's neighbors is equal to or greater than k. Jebara *et al.* [22] proposed a method to learn k-regular graphs, in which each node has exactly k neighbors. While obtain better robustness property, all of these methods have much higher complexity than kNN graph and ϵ graph because of the complex optimization problems involved. Xie *et al.* [45] proposed to learning the normalized graph Laplacian matrix with the help of supervised information.

III. Minimum Tree Cut Method

A. Notations

Recall l is the number of labeled data, u is the number of unlabeled data, and $n = l + u$ is the size of the data set. K is the number of classes.

We use G to denote a graph, T to denote a tree, while $E(G)$ and $E(T)$ denote the edge set of graphs G and tree T, respectively. w_{ij} describes the weight for edge (i, j). For a tree, $\uparrow(i)$ denotes the parent of node i, $\downarrow(i)$ denotes the set of i's children, and $\downarrow^*(i)$ denotes the set of nodes in the subtree rooted at i. $|S|$ is the size of set S. A cut is an edge connecting two nodes that have different labels. Given a labeling $\mathbf{f} \in \{1, \ldots, K\}^n$ of a graph G, the cut size of $\mathbf{f}$ on G is the sum of weights of all cuts, which can be denoted as $\sum_{(i,j)\in E(G)} w_{ij} I\{f_i \neq f_j\}$.

B. Proposed Method

Blum and Chawla [4] and Blum *et al.* [5] first proposed to predict the graph with the labeling that minimizes the cut size of the graph, and meanwhile is consistent with all the labeled data. However, for multiclass discrete variables, this optimization problem is Non-deterministic Polynomial-time-hard. Thus, Zhu *et al.* [51] binarized the problem via a standard one-vs-all scheme, and solved it in the continuous space. However, their method is still too expensive to solve large-scale problems.

Inspired by the idea of graph sparsification [9], [21], we propose a two-step TL method, called MTC, in this paper: first, we approximate the graph by a spanning tree to simplify the graph structure. Then, we solve the constrained cut-size minimization problem directly on the discrete label set. Formally, after generating a spanning tree, our method labels the nodes by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{f}} \quad & \sum_{(i,j)\in E(T)} w_{ij} I\{f_i \neq f_j\} \\ \text{s.t.} \quad & f_i = y_i \quad i = 1, \ldots, l \\ & \mathbf{f} \in \{1, \ldots, K\}^n. \end{aligned} \tag{1}$$

As we will show, by exploiting the special structure of tree, the above problem can be exactly solved in linear time by a dynamic programming algorithm.

As pointed out in [4] and [26], mincut problems may have multiple solutions in theory. But in practice, pairwise distances of high-dimensional data points are usually different from each other, which results in different edge weights given by $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2\}$. When weights of edges are different, it is very unlikely to have multiple solutions. This phenomenon has actually been observed in our experiments.

In the next two sections, we first introduce the tree labeling algorithm that solves (1); then discuss possible ways to generate a spanning tree from a graph.

C. Tree Labeling Algorithm

Before detailing the algorithm, we first define some concepts that will be used extensively in our method. The first one, called *label-bordered tree* (lb-tree), was first introduced in [9]. Formally, given a partially labeled tree T, an lb-tree is any maximal subtree of T whose leaves are all labeled and no internal node is labeled. The second concept is called *single-color tree* (sc-tree). Given all the lb-trees, an sc-tree is defined as any maximum subtree sharing one and only one common node with one single lb-tree. Moreover, we call the common node as the connecting node. Obviously, different sc-trees have no common nodes. Furthermore, we say an unlabeled node is internal if it is in a certain lb-tree, otherwise it is external (Fig. 1).

Our tree labeling algorithm proceeds as follows. First, it splits a partially labeled tree T into label-bordered subtrees (lb-subtrees) and single-color trees (sc-subtrees). Then, it use a procedure to label each lb-subtree. Finally, all the nodes in one sc-subtree are labeled by its connecting node's label.

To split a partially labeled tree, we first classify unlabeled nodes into internal and external categories. This can be done by a post-order visit of the tree from any labeled node. An unlabeled node is internal, if and only if it has at least one child that is internal or labeled; otherwise, it is external.

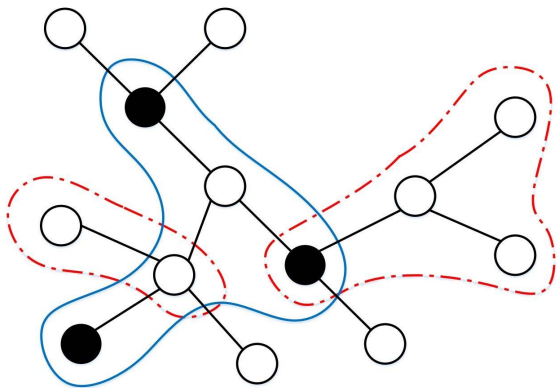


Fig. 1. Illustration of lb-tree and sc-tree. In this partially labeled tree, black nodes denote labeled data, and blank nodes are unlabeled. The subtrees in solid lines are lb-trees, and the subtrees in dash lines are sc-trees.

After that, the tree splitting can be done by calling the *split(r)* function in Algorithm 1. For succinctness, we briefly list the procedure without specific explanation. Since it visits each edge at most once, its computational cost is $O(n)$.

In the following, we design another procedure to label an lb-tree. This is one major contribution of our work. Given an lb-tree T, we predict its unlabeled nodes in such a way that the cut size of T is minimized. The core of the procedure is to compute the function $cutsize(i,k)$ which is defined as the minimum cut size of the subtree rooted at node i when i is labeled as k. Because of the special structure of tree, this value can be calculated directly by the $cutsize$ values of node i's children. Specifically, for an unlabeled node i in an lb-tree, we have

$$\begin{aligned}&\text{cutsize}(i,k)\\&=\sum_{j\in\downarrow(i)}\min\{\text{cutsize}(j,k),\min\{\text{cutsize}(j,\tilde{k})+w_{ij}:\tilde{k}\neq k\}\}\\&=\sum_{j\in\downarrow(i)}\min\{\text{cutsize}(j,k),\min\{\text{cutsize}(j,\tilde{k}):\forall\tilde{k}\}+w_{ij}\}.\end{aligned}\quad(2)$$

For leaf nodes, recall that the leaves of an lb-tree are all labeled. We define the $cutsize$ value of a leaf i as

$$\text{cutsize}(i,k)=\begin{cases}0, & k=y_i\\ \infty, & k\neq y_i.\end{cases}\quad(3)$$

Thus, the $cutsize$ values of nodes in the lb-tree can be computed from bottom to top by (2) and (3). To store all $cutsize$ values for an lb-tree, a table of size $|\downarrow^*(r)|\times K$ is needed. To compute the ith row of the table, we first compute $\min\{\text{cutsize}(j,k):\forall k\}$ for all $j\in\downarrow(i)$, which needs $O(|\downarrow(i)|\times K)$ operations. Then, for each k, $\text{cutsize}(i,k)$ can be computed in $O(|\downarrow(i)|)$. Thus, it needs $O(|\downarrow(i)|\times K)$ operations to fill one row, and $\sum_i|\downarrow(i)|\times K=|\downarrow^*(r)|\times K$ to compute the whole table. Furthermore, the procedure needs $O(Kn)$ space and $O(Kn)$ operations to compute all lb-trees's $cutsize$ values.

Once the $cutsize$ values of the root r are determined, by definition $\min\{\text{cutsize}(r,k):k=1,\ldots,K\}$ is the minimum cut size of the lb-tree. We can then predict each unlabeled node to achieve this minimum cut size. This is done by a traversal from top to bottom. For the root node r, we predict it by

$$k^*(r)=\arg\min_k\{\text{cutsize}(r,k)\}.$$

For any other unlabeled node i, suppose we have predicted its parent's label as $k^*(\uparrow(i))$. We predict i as $k^*(i)$ that is calculated by

$$k^*(i)=\begin{cases}k^*(\uparrow(i)) & \text{if } \text{cutsize}(i,k^*(\uparrow(i)))\\ & \le\text{cutsize}(i,\tilde{k}(i))+w_{i\uparrow(i)}\\ \tilde{k}(i) & \text{otherwise}\end{cases}$$

where $\tilde{k}(i)=\arg\min_k\{\text{cutsize}(i,k):k\neq k^*(\uparrow(i))\}$. Since the prediction for each node requires K comparisons, the computational cost for predicting all lb-trees is $O(Kn)$.

Now, we formally prove that the proposed algorithm exactly solves the optimization problem (1).

Theorem 1: The tree labeling algorithm introduced in Section III-C optimizes (1) exactly.

Algorithm 1 Tree Splitting Procedure

```
Function: split_internal(i)
begin
    children ← ∅;
    for each j ∈↓ (i) do
        if j is an external node then
            output {i, ↓* (j)} as a sc-tree;
        else if j is a labeled node then
            push(j);
            children ← children⋃{j};
        else
            children ← children⋃split_internal(j);
    return: children ⋃ {i};
end

Function: split_labeled(i)
begin
    for each j ∈↓ (i) do
        if j is an external node then
            output {i, ↓* (j)} as a sc-tree;
        else if j is a labeled node then
            push(j);
            output {i, j} as an lb-tree;
        else
            children ← split_internal(j);
            output {i, children} as an lb-tree;
end

Function: split(r)    //r must be a labeled node
begin
    stack← ∅;
    push(r);
    while stack≠ ∅ do
        j ← pop();
        split_labeled(j);
end
```

Proof: We denote the optimal value of (1) as mincut$(T, \mathbf{y})$. It is the minimum cut size on a tree T with respect to the given labels $\mathbf{y}$ on that tree. Because of the special structure of tree, we have

$$\begin{aligned}&\text{mincut}(T, \mathbf{y})\\ &= \sum_{T_i \in \texttt{sc-tree}(T)} \text{mincut}(T_i, \mathbf{y}_i) + \sum_{T_i \in \texttt{lb-tree(T)}} \text{mincut}(T_i, \mathbf{y}_i)\\ &\geq \sum_{T_i \in \texttt{lb-tree(T)}} \text{mincut}(T_i, \mathbf{y}_i)\end{aligned}$$

where sc-tree(T) and lb-tree(T) are the sets of the sc-trees and the set of the lb-trees of T. On the other hand, suppose the solution given by the algorithm is $\mathbf{f} \in \{1, \ldots, K\}^n$. By construction, $\mathbf{f}$ induces no cut on any sc-tree and achieves the minimum cut size on each lb-tree. Thus, the cut size of $\mathbf{f}$ on the whole T is exactly $\sum_{T_i \in \texttt{lb-tree(T)}} \text{mincut}(T_i, \mathbf{y}_i)$. This completes our proof. □

To conclude this part, we emphasize that, since each step of the tree labeling algorithm (tree splitting cutsize computing and labeling) has a time and space cost of $O(n)$, its overall complexity is $O(n)$.

D. Generate a Spanning Tree From a Graph

To apply the tree labeling algorithm introduced above to the general transductive classification problem, one has to construct a graph G from the data set, and then generates a spanning tree T from G. We have introduced existing graph construction methods in Sections I and II. In this section, we briefly discuss typical methods for generating spanning trees.

MST is a spanning tree that minimizes the total cost. It is easy to see MST is the solution of $\max\{\sum_{(i,j)\in E(T)} w_{ij} : T \in \mathcal{T}(G)\}$, where $\mathcal{T}(G)$ is the set of spanning trees of graph G. MST can be generated by Kruskal's algorithm in $O(|E| \log n)$ time.

Shortest path tree (SPT) is a spanning tree that the distance between a selected node and all other nodes is minimal. Herbster *et al.* [21] used this tree as an approximate solution of $\min\{\text{tr}(T^+ - G^+) : T \in \mathcal{T}(G)\}$, where tr$(A)$ denotes the trace of A, and A^+ denotes the pseudoinverse of A. SPT can be generated by Dijkstra's algorithm in $O((|E| + n) \log n)$ time.

Random spanning tree (RST) is a spanning tree taken with the probability proportional to the product of its edge weights. Cesa-Bianchi *et al.* [10] used this tree to ensure a worst-case bound of their online graph labeling algorithm. Another advantage of RST is that it can be generated in $O(n)$ time for most cases.

In case that the graph is not connected, we can run the tree labeling algorithm on each connected component, respectively. To further boost the performance, one can use the ensemble method. Specifically, one can generate multiple spanning trees, run the tree labeling algorithm on each tree, and then use the majority vote method to predict unlabeled nodes. As we will show in the following, this approach can significantly improve the accuracy.

IV. Insensitiveness to Graph Construction

One major problem of graph-based TL is that their performance is sensitive to the graph construction. For the kNN graph or ϵ graph, a small perturbation in k, ϵ, or σ^2 can result in big fluctuations in accuracy.

In this section, we formally show that the structure of MST is insensitive to these parameters. This motivates us to use MST to approximate the graph, which makes the whole MTC method robust to graph construction. Experimental results in Section V will further validate this desirable property.

First, we prove that the structure of MST is invariant to the ϵ graph theoretically.

Proposition 1: Suppose $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n : \mathbf{x}_i \in \mathbb{R}^d\}$ is a set of data points. G and G' are two connected graphs built on S by the ϵ-graph method with different ϵ, and $E(G)$, $E(G')$ are weighted by $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2\}$ with same σ. For each edge e_{ij}, let its cost $\pi_{ij} = -w_{ij}$. If T and T' are MSTs of G and G', respectively, then $E(T) = E(T')$.

Proof: To prove this proposition, we first recall Kruskal's MST algorithm that proceeds as follows. It first sorts all the edges into an increasing order by their costs; then, the next lightest edge producing no cycle is repeatedly added.

Without loss of generality, we assume $\epsilon < \epsilon'$. Then, we have: 1) $E(G) \subseteq E(G')$ and 2) for $\forall e \in E(G)$ and $\forall e' \in E(G') \setminus E(G)$, $\pi_e < \pi_{e'}$. Thus, edges that are in $E(G')$ but not in $E(G)$ are behind the edges in $E(G)$ in the sorted edge sequence of G'. Therefore, the procedures of Kruskal's algorithm for G and G' are the same that implies $E(T) = E(T')$. □

When the graph is constructed by kNN, simple examples can be constructed to show MST is not robust to k in the mathematical sense. But still, we observe this robust property in practical experiments.

Next, we prove that the structure of MST is invariant to the RBF weighting function's parameter σ.

Proposition 2: Suppose G and G' are two connected graphs built on a data set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n : \mathbf{x}_i \in \mathbb{R}^d\}$. Furthermore, G and G' have the same edge set $E(G) = E(G')$, which are weighted by $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2\}$ with different σ. For each edge e_{ij}, let its cost $\pi_{ij} = -w_{ij}$. Then, if T and T' are MSTs of G and G', respectively, then $E(T) = E(T')$.

Proof: The proof is similar to the above one. Since $E(G) = E(G')$ and the RBF weighting function is monotonic with respect to $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, the sorted edge sequences of G and G' are exactly the same. Therefore, the procedures of Kruskal's algorithm for G and G' are same that implies $E(T) = E(T')$. □

Roughly speaking, Proposition 1 states the edge set of MST is robust to ϵ in the ϵ-nearest-neighbor graph construction method, while Proposition 2 states that the edge set of MST is robust to σ.

V. Experiments

We conduct extensive experiments to evaluate our method in accuracy, robustness, and speed. In addition, we examine the performance of different types of spanning trees and the

TABLE I
BRIEF DESCRIPTION OF DATA SET

Data Set	Size	Dim	Class
Sonar	208	60	2
Ionosphere	351	34	2
Breast Cancer	683	10	2
COIL20	1440	1024	20
USPS	9298	256	10
MNSIT	70000	784	10
RCV1	9625	29992	4
Newsgroup20	19996	1355191	2

effect of ensemble of multiple trees. Applications for large-scale problems are left in the next section. All the experiments are conducted on a PC with a 3.10-GHz 4-core CPU and 8-GB RAM. Our codes and some of the data sets are available at https://sourceforge.net/projects/minimutreecut/files/.

A. Data Set

We used eight popular data sets from different fields to evaluate our method.[1] In the following, we give an introduction to these data sets, and summarize their basic property in Table I.

1) *UCI Data Set:*
   a) Sonar, Ionosphere, and Breast Cancer are three popular used data sets for SSL that are taken from UCI data repository. No preprocessing is performed.
2) *Image:*
   a) COIL20 is an image data set that contains 20 objects. The images of each object were taken 5° apart and each object has 72 images. The size of each image is 32×32 pixels, with 256 gray levels per pixel.
   b) USPS and MNIST are two handwritten digit data sets. USPS contains $7291 + 2007 = 9298$ gray images of size 16×16, while MNIST contains $60\,000 + 10\,000 = 70\,000$ gray images of size 28×28.
3) *Text:*
   a) RCV1 is a subset of the RCV1-v2 corpus that is a collection of Reuters news stories. A set of 9625 documents with 29 992 distinct words is chosen, including categories C15, ECAT, GCAT, and MCAT. The standard term frequency - inverse document frequency (TF-IDF) method is used to extract the feature vector for each document. See [8] for details.
   b) Newsgroup20 is a two-class variant of the 20 Newsgroups data set. The positive class consists of the 10 groups with names of form sci.*, comp.*, or misc.forsale, and the negative class consists of the other 10 groups. After preprocessing, the data set has 19 996 documents with 1 355 191 distinct words. We used binary term frequencies and normalized each example vector to unit length. See [25] for details.

[1]The UCI data sets are from archive.ics.uci.edu/ml/. The COIL20, USPS, and RCV1 are from www.cad.zju.edu.cn/home/dengcai/Data/data.html. The MNIST data set is from yann.lecun.com/exdb/mnist/. The Newsgroup20 is from www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

B. Graph Construction

Without particular claim, we construct kNN graphs in all the experiments, as kNN graph is superior to ϵ graph on most cases. Different dissimilarity functions are adapted to determine the kNNs for different data sets. For UCI and image data sets, we use $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, while use $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - < \mathbf{x}_i, \mathbf{x}_j > / \|\mathbf{x_i}\| \|\mathbf{x_j}\|$ for text data sets. We use the RBF kernel function $\exp\{-d(\mathbf{x}_i, \mathbf{x}_j)/\sigma_0^2\}$ to weight the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ with $\sigma_0^2 = \sum_{(i,j)\in E} d(\mathbf{x}_i, \mathbf{x}_j)/|E|$. Note by theory most pairwise distances in high-dimensional space are similar, thus in general traditional distances are invalid for measuring the data similarity. But for graph construction, we only use distances within small local regions in which they are still informative.

C. Accuracy

In this section, we evaluate our method's classification accuracy by comparing it with typical existing methods. We compare our method with two graph-based transductive methods: graph mincut (GCUT) [4], Gaussian random fields (GRFs) [51], and learning local and global consistency (LLGC) [48], one tree-based method: the graph perceptron algorithm (GPA) [21], and one line-based method: the weighted tree algorithm (WTA) [10].[2]

For each data set, we randomly select l data points as labeled data and perform different algorithms to predict the unlabeled data. To control the variance of results, we repeat the procedure 20 times for different labeled/unlabeled splits. For graph-based methods (GCUT, GRF, and LLGC), we manually choose k from $\{2, 4, 8, 16, 32\}$ such that the test error is minimized. For the tree-based method and the line-based method, we directly fix k to 32, because they are insensitive to this parameter (as validated theoretically in Section IV and empirically in the following section).

We let l vary from $n \times \{1\%, 2\%, 4\%, 8\%, 16\%, 32\%, 64\%\}$, and report the results in Fig. 2. As observed in the following parts, the performance of MST is always the best among different tree construction methods. Hence, we only report the classification results of different methods on MST.

As observed from the results, the following conclusions can be made.

1) The accuracy of GRF and LLGC is higher than the tree-based methods MTC and GPA and line-based method WTA. But when l becomes large, the gap becomes little.
2) To our surprise, the accuracy of GCUT is lower than MTC. When the number of labeled data is small, GCUT often gives highly unbalanced prediction.
3) MTC performs better than other tree-based and line-based methods in most cases. GPA also gets good accuracies, but as shown in the next section, GPA is much slower than our method. As analyzed before, WTA loses too much information in approximating a graph by a line, thus obtains bad performance.

[2]The name is because: 1) it is proposed by original authors and 2) the method generates a spanning tree as an intermediate step.

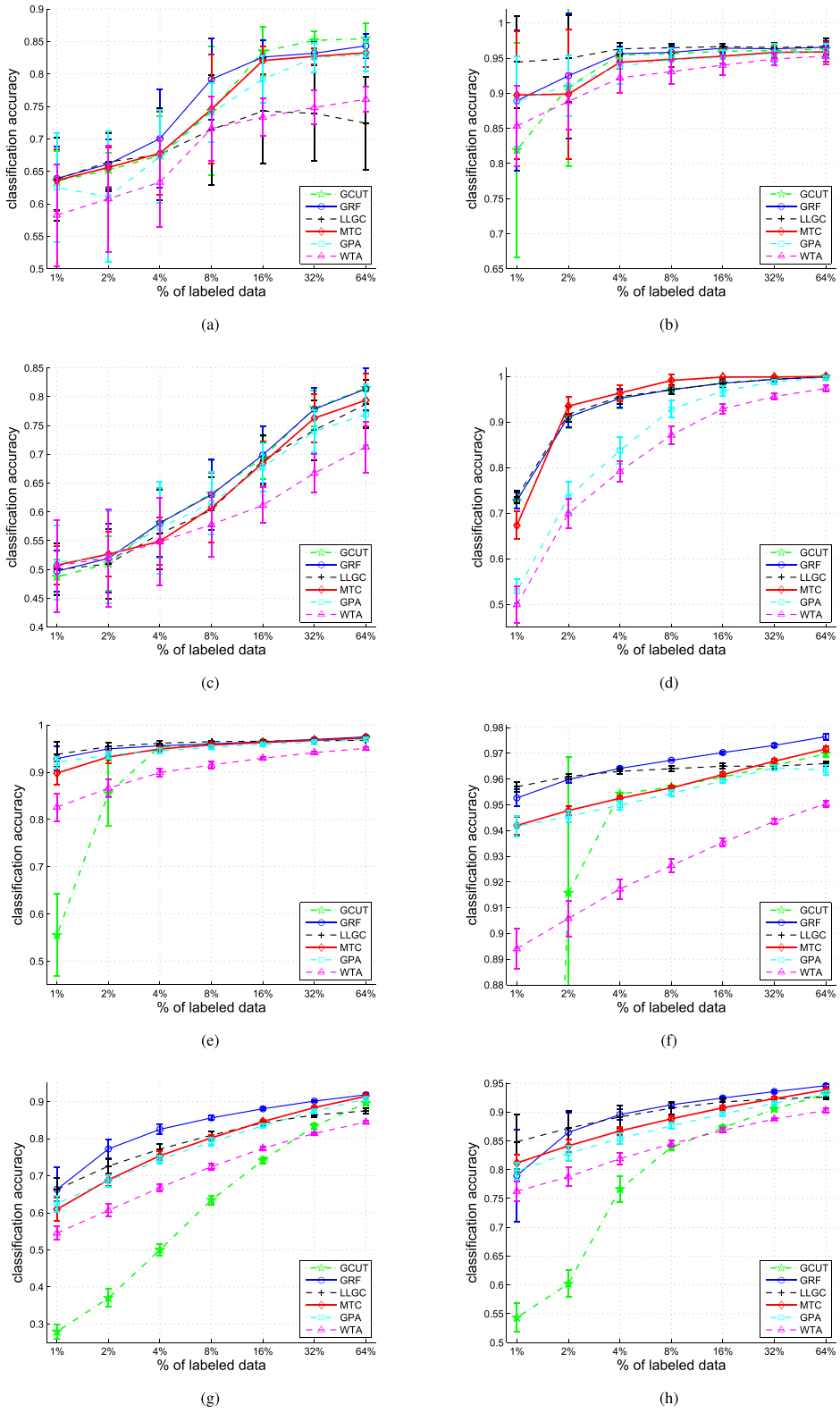


Fig. 2. Average classification accuracy for different labeled data size. The solid curves are results of graph-based methods, while the dashed ones are for tree- and line-based methods. (a) Ionosphere. (b) Breast Cancer. (c) Sonar. (d) COIL20. (e) USPS. (f) MNIST. (g) RCV1. (h) Newsgroups20.

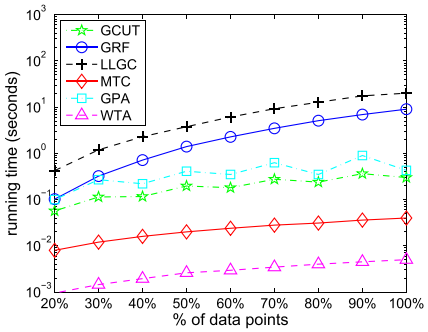
D. Speed

In this section, we perform three experiments to empirically compare the running time of different methods. All experiments in this section use 8NN graphs. To control the variance of results, we repeat each experiments 20 times for different labeled/unlabeled splits, and report the average running time.

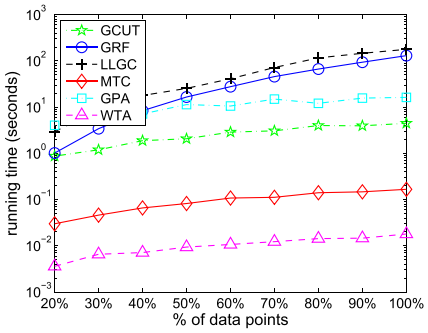
The first experiment proceeds as follows. We construct one 8NN graph for a data set, randomly select $l = 2\% \times n$ data

TABLE II
AVERAGE RUNNING TIME (IN SECONDS) ON FOUR DATA SETS

	USPS	RCV1	Newsgroups20	MNIST
GCUT	0.52	0.21	0.31	4.89
GRF	0.18	0.52	9.07	130.29
LLGC	0.69	1.28	20.10	178.43
MTC	0.02	0.02	0.04	0.17
GPA	0.38	0.27	0.43	16.30
WTA	0.00	0.00	0.01	0.02
graph building	5.05	6.83	52.85	503.52



(a)



(b)

Fig. 3. Average running time for different graph size on (a) Newsgroup20 and (b) MNIST. Note the y-axis is with logarithmic scale.

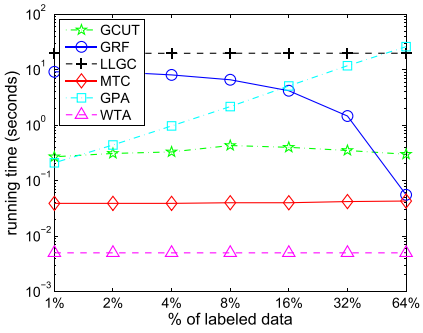
points as labeled data and perform different algorithms to predict the unlabeled data. The average running time on the four largest data sets is reported in Table II. We also include the running time for building 8NN graphs.

To observe the running time variation with the size of graph, we have designed the second experiment. We construct graphs of size $n \times \{20\%, 40\%, 60\%, 80\%, 100\%\}$ by subsampling from the whole data set. For each graph, we randomly select $l = 2\% \times n$ data points as labeled data and perform different algorithms to predict the unlabeled data. The average running time on newsgroup20 and MNIST is reported in Fig. 3. Note the y-axis is with logarithmic scale.

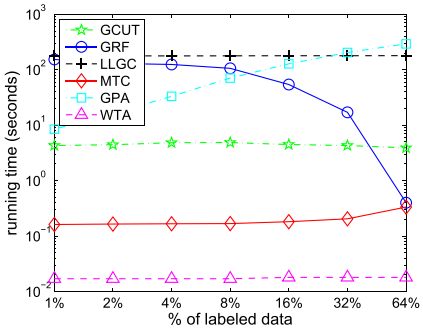
The last experiment is to examine methods' running time variation with the size of labeled data. For a given graph, we varies l from $n \times \{1\%, 2\%, 4\%, 8\%, 16\%, 32\%, 64\%\}$. The average running time on newsgroup20 and MNIST is reported in Fig. 4. Note the y-axis is with logarithmic scale.

We have the following observations.

1) The running time of MTC scales almost linearly with n, and almost invariant with l.



(a)



(b)

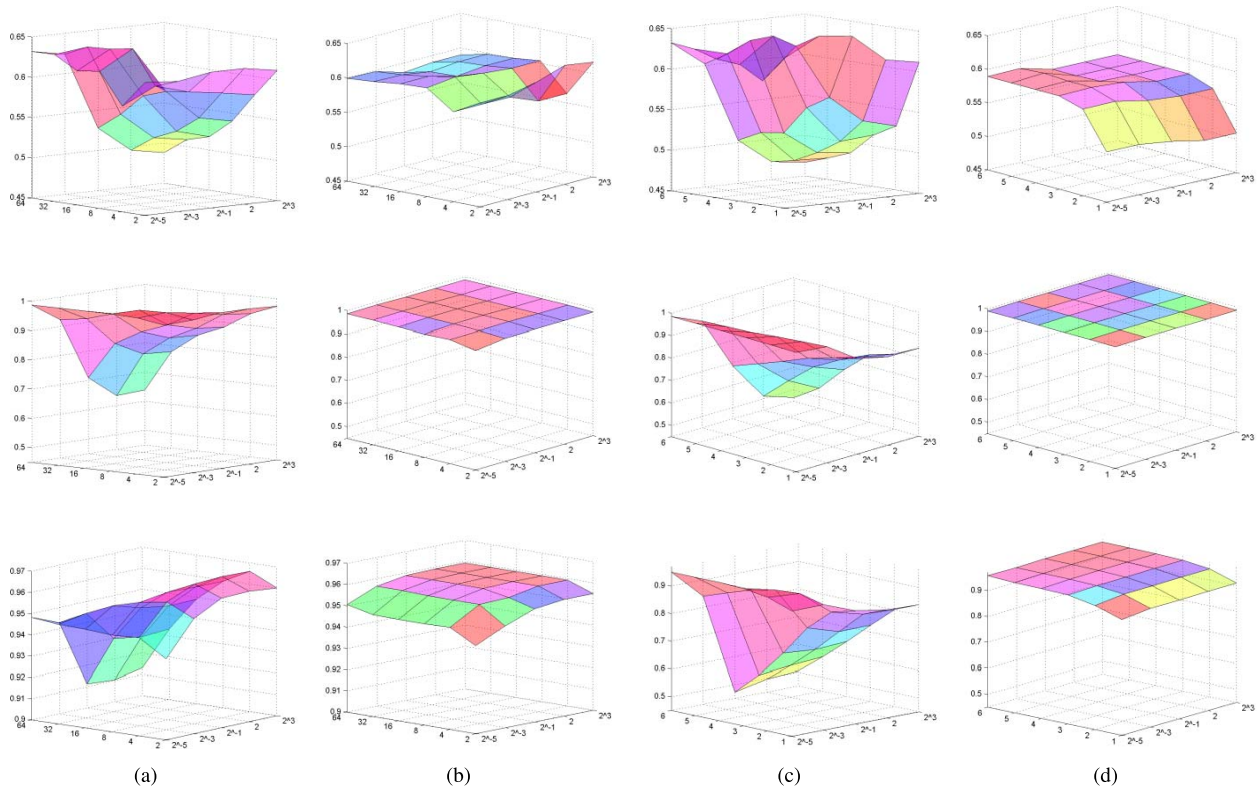
Fig. 4. Average running time for different labeled data size on (a) Newsgroup20 and (b) MNIST. Note the y-axis is with logarithmic scale.

2) Although all graph-based methods (GCUT, GRF, and LLGC) scale polynomially with n, their actual running time are very different. GCUT is faster than GRF and LLGC in most cases, but still much slower than MTC. On the MNIST data set, for $l = 2\%n$, MTC is 28 times faster than GCUT and 700 times faster than GRF and LLGC.
3) GPA is much more time consuming than MTC and WTA. On MNIST, our method is 100 times faster than GPA for $l = 2\%n$, while is 1000 times faster than GPA for $l = 32\%n$. This is because: 1) as the complexity of GPA is $O(ln)$, when l is comparable with n, its complexity is about $O(n^2)$ and 2) for some data sets, such as MNIST, GPA needs a large number of iterations to converge.
4) The WTA algorithm is even faster than MTC, but its accuracy is low (see the previous section).

E. Robustness

One major problem of graph-based TL is that their performance is sensitive to the graph construction. In this section, we validate the theory that using the MST to approximate a graph can solve this problem effectively.

We first construct kNN graphs and ϵ graphs with different parameters. Specifically, for kNN graphs, we vary k from $\{2, 4, 8, 16, 32, 64\}$. For ϵ graphs, we first separate the data points into 50 groups using Kmeans. Then, we compute the average pairwise distance within each cluster $\epsilon_0 = 1/50 \sum_{k=1}^{50} 1/n_k \sum_{i \in C_k} \|\mathbf{x}_i - \mathbf{C}_k\|^2$, where n_k is the size of the cluster k and $\mathbf{C}_k$ is the center of the cluster k.



(a) (b) (c) (d)

Fig. 5. Average classification accuracy with different graph construction parameters. The four columns are results of (a) GRF on kNN graphs, (b) MTC on kNN graphs, (c) GRF on ϵ graphs, and (d) MTC on ϵ graphs. The three rows are results on Sonar, COIL20, and USPS, respectively.

Finally, we vary ϵ from $\epsilon_0 \times \{1, 2, 3, 4, 5, 6\}$. For both kNN graph and ϵ graph, we vary the kernel width parameter σ^2 from $\sigma_0^2 \times \{2^{-5}, 2^{-3}, 2^{-1}, 2^1, 2^3\}$. Thus, we get 30 kNN graphs and 30 ϵ graphs for each data set.

Next, we randomly labeled $l = n \times 10\%$ data points and perform GRF and MTC to predict the unlabeled data. We repeat the procedure 10 times for different labeled/unlabeled splits, and report the average accuracy in Fig. 5.

The four columns are results of GRF on kNN graphs, MTC on kNN graphs, GRF on ϵ graphs, and MTC on ϵ graphs. The three rows are results on Sonar, COIL20, and USPS, respectively.

As we can see, the performance of MTC is very robust to the variance of k, σ^2, and ϵ. Given fixed k or ϵ, the performance of MTC is almost invariant to the variance of σ^2. However, when k or ϵ is very small, the constructed graph may contain many disconnected components that have no labeled data on. In that case, the performance of MTC may be not good. Thus, in practice, we only need to choose a big k or ϵ to ensure the connectivity, and let MST filter out those irrelevant edges.

On the other hand, as the increase of k or ϵ, the constructed graph may contain a large number of irrelevant edges that may severely hurt the performance of GRF.

F. Effect of Different Spanning Tree and Ensemble of Multiple Spanning Trees

In this part, we explore the performance of different types of the spanning tree. Specifically, we compare three types of spanning trees: MST, SPT, and RST. Compared with graphs, trees' presentation ability is limited. Naturally, it is interesting to see if we can boost the MTCs performance by the idea of ensemble. The method is simple: after generating multiple spanning trees, we label each tree with MTC. Then, we make the final prediction using the major voting among these trees. Since we cannot generate multiple MSTs in general, we only apply this method to SPT and RST. Thus, we compare the following seven methods: MST, SPT, SPT10 (ensemble of 10 SPTs), SPT20 (ensemble of 20 SPTs), RST, RST10, and RST20. The detailed experiment setting is same as that in Section V-C.

The results are shown in Fig. 6. We can see that: 1) among the three single tree methods, MST is always the best except Ionosphere, while RST is always the worst except Breast Cancer; 2) the ensemble method is very effective, especially for RST; and 3) although it is hard to say which one is the best method among all the seven methods, MST and SPT20 can usually provide very competitive results. On the other hand, the running time for generating MST is typically 2–3 times of that for generating RST, while the running time for SPT is between them. Thus, a balance should be made between the accuracy and computational cost.

VI. Applications in Large-Scale Problems

We apply our method to two large-scale real-world problems that have drawn wide interests: web-spam detection and interactive image segmentation.

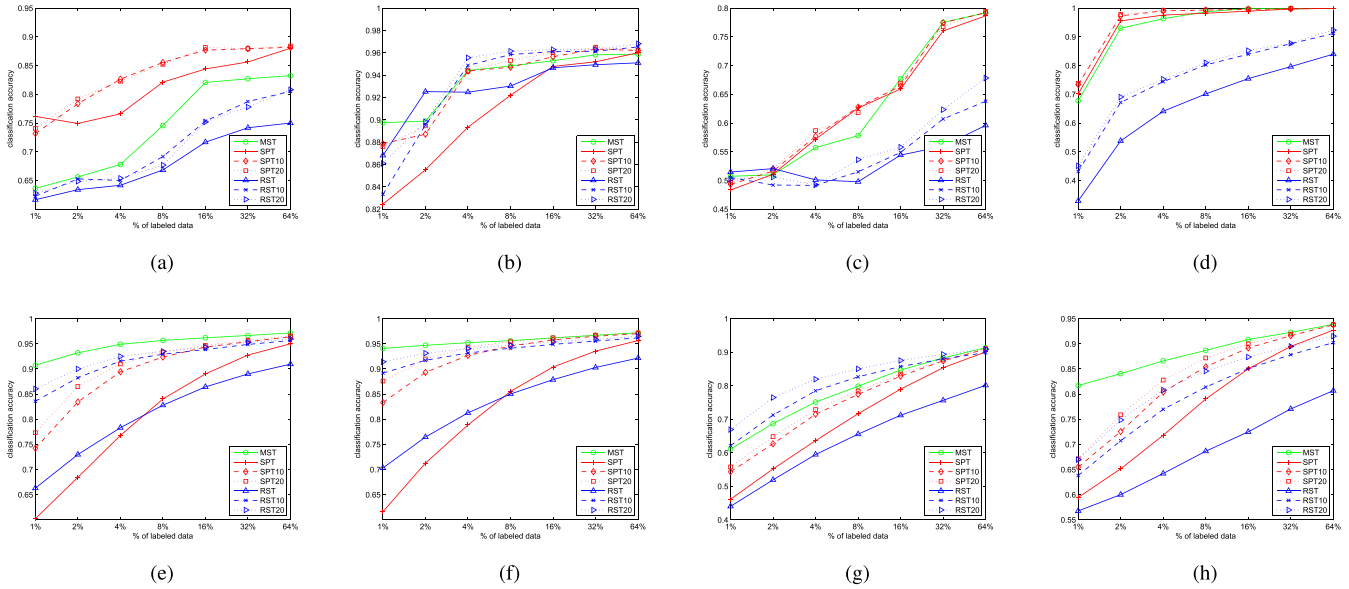


Fig. 6. Effect of a different spanning tree and ensemble of multiple spanning trees. The curves with different colors represent the results of different types of trees. The curves with different line shapes represent the results of ensembles of different number of trees. (a) Ionosphere. (b) Breast Cancer. (c) Sonar. (d) COIL20. (e) USPS. (f) MNIST. (g) RCV1. (h) Newsgroups20.

TABLE III
CLASSIFICATION ACCURACY ON WEB-SPAM DATA SET

Method	MTC	WTA	single GPA	ensemble GPA	Witshel et al.	Filoce et al.	Benczur et al.
Accuracy	99.6	99.0	97.6	99.1	99.5	99.4	94.2

A. Web-Spam Detection

We apply our method to the 2007 web-spam challenge developed by the University of Paris VI.[3] It contains 400 000 web pages. A web page is connected to another web page if there is at least one hyperlink from the former to the latter. Thus, the links are directed. Following [21], we discard directional information and assign a weight of 1 to unidirectional links and of 2 to the bidirectional links. This results in a graph with 400 000 nodes and 10 455 545 edges. Additional TF-IDF feature vectors of the web-pages' content are provided for each web page, but we have discarded this information. There are about 80% of nonspam web pages and 20% of spam ones. Following the published data set, we use 10% of labeled web page for training and 90% for testing.

Experimental results are shown in Table III. Due to the same experimental setup, the results except MTC and WTA are directly copied from [21]. For ensemble GPA, 21 MSTs are used, and predictions are made by majority voting. Witshel *et al.*, Filoce *et al.*, and Benczur *et al.* are three methods that participated the 2007 web-spam challenge. We did not report the result of graph-based methods as it is intractable to run them on this large data set. GPA takes about 30 min to train a single kernel perceptron. In comparison, our proposed MTC takes less than 10 s to complete the classification. Similarly, WTA is also very fast and finishes the learning also within 10 s. However, the classification accuracy of WTA is 99.0, which is lower than that of MTC. This once again demonstrates the advantages of our proposed method. The reported times include the time of generating MST and labeling the tree but do not include the time of loading graph from disk.

[3]http://webspam.lip6.fr/wiki/pmwiki.php

B. Interactive Image Segmentation

Segmenting foreground objects from natural images is a fundamental task in image understanding. However, it is very challenging because of the high complexity of visual patterns and the lack of semantic knowledge. To make the problem easier, one possible way is to use an interactive method, in which users are required to identify some parts of the image as foreground and background. Thus, this is exactly a TL setting, where the user-identified pixels are the labeled data and the other pixels are the unlabeled data to be classified.

A number of algorithms have been proposed [1], [7], [34], [35]. Although obtain satisfying performance, all these algorithms suffer from high computing complexity. Actually, the biggest image used to test algorithms in those papers is 640×480. However, nowadays most mobile phones can take picture with millions of pixels.

In this section, we apply the MTC method to deal with the interactive segmentation task on large images. For each image, we manually draw some red lines on the foreground object and green lines on the background. Then, we treat the pixels labeled by red lines as $+1$ class and pixels labeled by green lines as 0 class, and try to predict all the other unlabeled pixels. To construct the graph, we connect each pixel with pixels that are located in its 7×7 neighborhood. The edge's

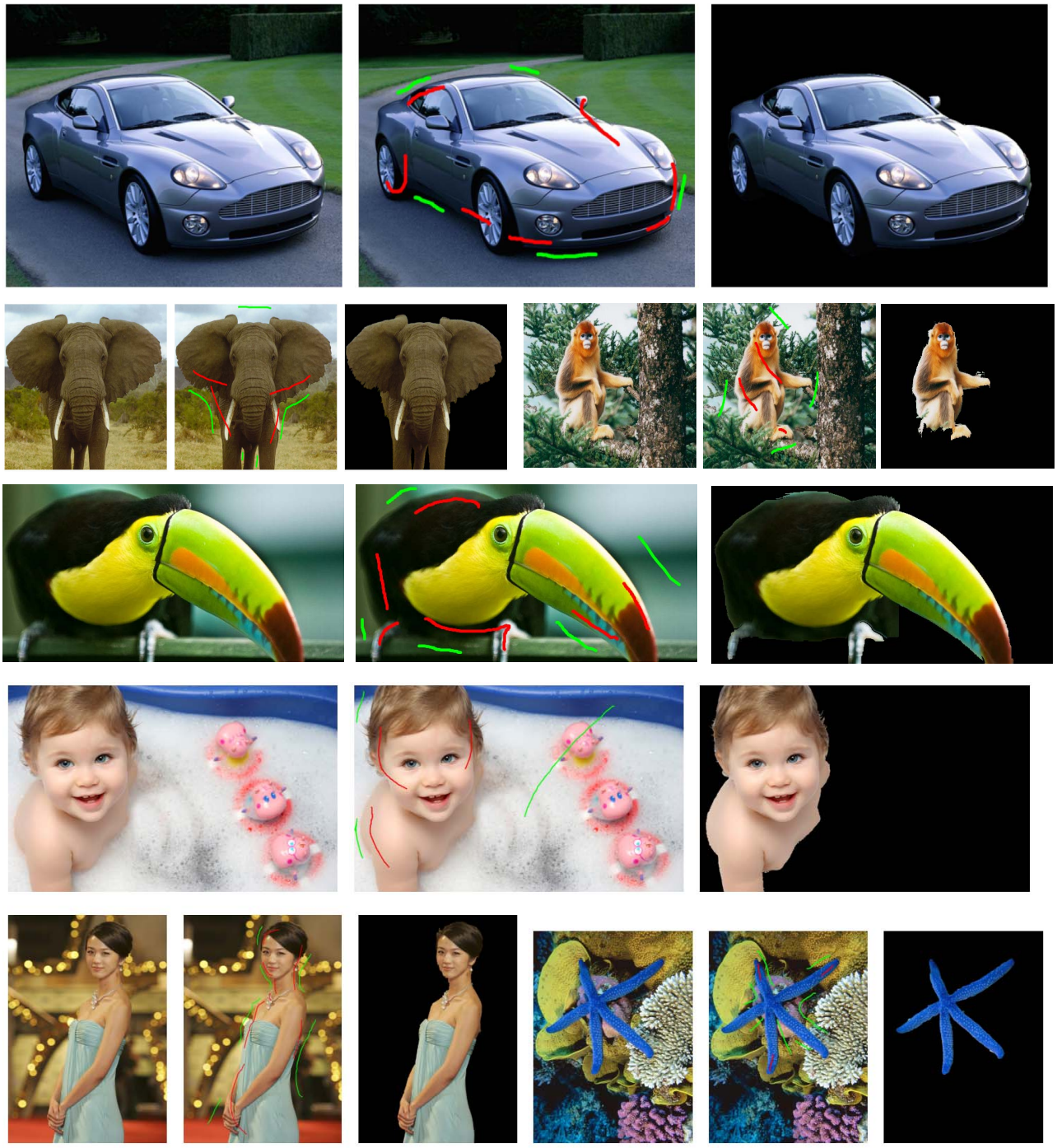


Fig. 7. Interactive image segmentation. In each image group, the left is the original image, the middle is the labeled image, and the right is the segmentation result.

TABLE IV
IMAGE SIZE AND RUNNING TIME FOR SEGMENTATION

Image	Car	Elephant	Monkey	Bird	Baby	Actress	Seafish
Size of image	800×600	1024×768	427×320	592×302	2560×1600	1171×1684	1200×1600
Running time (s)	1.9	3.7	0.6	1.0	71.2	13.5	23.4

weight is computed as $\exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/100\}$, where $\mathbf{x}_i$ is a three dimension vector contains the pixel i's red green blue value. One single MST is used.

The segmentation results and the corresponding running time are shown in Fig. 7 and Table IV, respectively. As we can see, MTC always provides nice results in short time. For example, the constructed graph for the baby image contains 4 096 000 nodes and 163 465 800 edges. The running time is about 70 s, which is even shorter than the graph construction time (about 117 s).

VII. Conclusion

Despite the great success of graph-based TL methods in the past decade, most of them have problems in scalability and robustness. With the aim of overcoming these problems, we proposed a simple but very practical graph-based TL algorithm in this paper. Inspired by previous works [9], [21], our method first approximates a graph by a spanning tree, and then labels the tree by solving a constrained MTC problem. Compared with traditional graph-based methods, our method has three outstanding characteristics: 1) it is efficient in speed and ready for large-scale applications; 2) it is robust to the graph construction parameters; and 3) it has no hyperparameter need to tune. Compared with other scalable algorithms, empirical results show our method consistently obtains better accuracy.

Like all cut-based methods, MTC may produce unbalanced results. For continuous-relaxed methods, even results can be obtained by adding the constraint $\mathbf{f}^T\mathbf{1} = 0, \mathbf{f}_i \in [-1, +1]$ or performing a post-processing as in [51]. But for discrete optimization problem where $\mathbf{f}_i \in \{-1, +1\}$, this becomes much harder. In the future, we will work on this problem. Another potential problem is that MTC places hard constraints on the labeled data, which makes it sensitive to incorrectly labeled data. We will explore the possibility of using soft constraints in the future.

References

[1] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[2] M. Belkin and P. Niyogi, "Semi-supervised learning on Riemannian manifolds," *Mach. Learn.*, vol. 56, nos. 1–3, pp. 209–239, 2004.

[3] 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.

[4] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 19–26.

[5] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy, "Semi-supervised learning using randomized mincuts," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 13.

[6] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.

[7] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proc. 8th Int. Conf. Comput. Vis.*, 2001, pp. 105–112.

[8] D. Cai and X. He, "Manifold adaptive experimental design for text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 707–719, Apr. 2012.

[9] N. Cesa-Bianchi, C. Gentile, and F. Vitale, "Fast and optimal prediction on a labeled tree," in *Proc. Conf. Learn. Theory*, 2009, pp. 145–156.

[10] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella, "Random spanning trees and the prediction of weighted graphs," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 175–182.

[11] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *J. Mach. Learn. Res.*, vol. 9, pp. 203–233, Jun. 2008.

[12] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. Int. Workshop Artif. Intell. Statist.*, 2005.

[13] J. Chen, H.-R. Fang, and Y. Saad, "Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection," *J. Mach. Learn. Res.*, vol. 10, pp. 1989–2012, Dec. 2009.

[14] B. Cheng, J. Yan, S. Yan, Y. Fu, and T. S. Huang, "Learning with ℓ^1-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.

[15] S. I. Daitch, J. A. Kelner, and D. A. Spielman, "Fitting a graph to vector data," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 201–208.

[16] S. Dasgupta, M. L. Littman, and D. McAllester, "PAC generalization bounds for co-training," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2002.

[17] O. Delalleau, Y. Bengio, and N. Le Roux, "Efficient non-parametric function induction in semi-supervised learning," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 96–103.

[18] W. Dong, C. Moses, and K. Li, "Efficient K-nearest neighbor graph construction for generic similarity measures," in *Proc. Int. Conf. World Wide Web*, 2011, pp. 577–586.

[19] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA, USA: MIT Press, 2005.

[20] M. Herbster, G. Lever, and M. Pontil, "Online prediction on large diameter graphs," in *Advances in Neural Information Processing Systems 21*. Red Hook, NY, USA: Curran Associates, 2008.

[21] M. Herbster, M. Pontil, and S. R. Galeano, "Fast prediction on a tree," in *Advances in Neural Information Processing Systems 21*. Red Hook, NY, USA: Curran Associates, 2008.

[22] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 441–448.

[23] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 200–209.

[24] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 290–297.

[25] S. S. Keerthi and D. DeCoste, "A modified finite Newton method for fast solution of large scale linear SVMs," *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 341–361, 2005.

[26] J. Kleinberg and É. Tardos, *Algorithm Design*. Delhi, India: Pearson Education, 2006.

[27] N. D. Lawrence and M. I. Jordan, "Semi-supervised learning via Gaussian processes," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA, USA: MIT Press, 2005.

[28] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 679–686.

[29] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proc. IEEE*, vol. 100, no. 9, pp. 2624–2638, Sep. 2012.

[30] G. S. Mann and A. McCallum, "Simple, robust, scalable semi-supervised learning via expectation regularization," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 593–600.

[31] D. J. Miller and H. S. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *Advances in Neural Information Processing Systems 9*. Cambridge, MA, USA: MIT Press, 1997.

[32] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proc. 9th Int. Conf. Inf. Knowl. Manage.*, 2000, pp. 86–93.

[33] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 103–134, 2000.

[34] A. Protiere and G. Sapiro, "Interactive image segmentation via adaptive weighted distances," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1046–1057, Apr. 2007.

[35] C. Rother, V. Kolmogorov, and A. Blake, "'GrabCut': Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

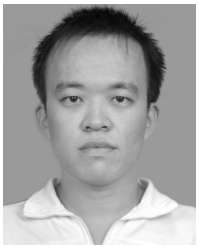
[36] A. Saluja and B. Kveton, "Semi-supservised learning with cover trees," in *Proc. Syst. Workshop Parallel Large-Scale Mach. Learn.*, 2011.

[37] A. Sarkar, "Applying co-training methods to statistical parsing," in *Proc. 2nd Meeting North Amer. Chapter Assoc. Comput. Linguistics Lang. Technol.*, 2001, pp. 1–8.

[38] B. M. Shahshahani and D. A. Landgrebe, "The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 5, pp. 1087–1095, Sep. 1994.

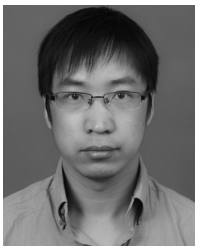
[39] V. Sindhwani and P. Niyogi, "A co-regularization approach to semi-supervised learning with multiple views," in *Proc. ICML Workshop Learn. Multiple Views*, 2005, pp. 74–79.

[40] D. A. Spielman and S.-H. Teng, "Solving sparse, symmetric, diagonally-dominant linear systems in time $O(m^{1.31})$," in *Proc. 44th Annu. IEEE Symp. Found. Comput. Sci.*, 2003, pp. 416–427.
[41] A. Talwalkar, S. Kumar, and H. Rowley, "Large-scale manifold learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[42] F. Vitale, N. Cesa-Bianchi, C. Gentile, and G. Zappella, "See the tree through the lines: The Shazoo algorithm," in *Advances in Neural Information Processing Systems 24*. Red Hook, NY, USA: Curran Associates, 2011.
[43] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[44] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-NN graph construction for visual descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1106–1113.
[45] B. Xie, M. Wang, and D. Tao, "Toward the optimization of normalized graph Laplacian," *IEEE Trans. Neural Netw.*, vol. 22, no. 4, pp. 660–666, Apr. 2011.
[46] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1233–1240.
[47] Y.-M. Zhang, K. Huang, and C.-L. Liu, "Fast and robust graph-based transductive learning via minimum tree cut," in *Proc. 11th Int. Conf. Data Mining*, Dec. 2011, pp. 952–961.
[48] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16*. Cambridge, MA, USA: MIT Press, 2004.
[49] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
[50] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2008.
[51] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.



Yan-Ming Zhang received the bachelor's degree from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, in 2011.

He is currently an Assistant Professor with the National Laboratory of Pattern Recognition, CASIA. His current research interests include machine learning and pattern recognition.



Kaizhu Huang (M'09) received the B.S. degree in automation from Xi'an Jiaotong University, Xi'an, China, the M.S. degree in pattern recognition from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, and the Ph.D. degree from the Chinese University of Hong Kong (CUHK), Hong Kong, in 2004.

He was a Researcher with the Fujitsu Research and Development Centre, CUHK, and the University of Bristol, Bristol, U.K., until 2009. He was also an Associate Professor with CASIA from 2009 to 2012. Since 2012, he has been an Associate Professor and the Director of the Multimedia Telecommunications Program with Xi'an Jiaotong-Liverpool University, Suzhou, China. He is involved in machine learning and its applications to pattern recognition. He has authored over 80 international papers, including 26 SCI-indexed journals and many first-tier conference papers, e.g., the Conference on Neural Information Processing Systems, the International Joint Conference on Artificial Intelligence, the Conference on Computer Vision and Pattern Recognition, the Conference on Uncertainty in Artificial Intelligence, and the International Conference on Data Mining.

Guang-Gang Geng received the Ph.D. degree from the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

He was with the China Internet Network Information Center, Beijing, in 2008. He is currently an Associate Professor with the Computer Network Information Center, Chinese Academy of Sciences. His current research interests include machine learning, adversarial information retrieval on the Web, and Web search.

Cheng-Lin Liu (SM'04) received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, in 1989, the M.E. degree in electronic engineering from Beijing Polytechnic University, Beijing, China, in 1992, and the Ph.D. degree in pattern recognition and intelligent control from the Chinese Academy of Sciences, Beijing, in 1995.

He was a Post-Doctoral Fellow with the Korea Advanced Institute of Science and Technology, Daejeon, Korea, and the Tokyo University of Agriculture and Technology, Tokyo, Japan, from 1996 to 1999. From 1999 to 2004, he was a research staff member and Senior Researcher with the Central Research Laboratory, Hitachi, Ltd., Tokyo. He is currently a Professor and the Director of the National Laboratory of Pattern Recognition with the Institute of Automation, Chinese Academy of Sciences. He has authored over 130 technical papers at prestigious international journals and conferences. His current research interests include pattern recognition, image processing, neural networks, machine learning, and in particular, the applications to character recognition and document analysis.