Fast and Robust Graph-based Transductive Learning via Minimum Tree Cut

Yan-Ming Zhang, Kaizhu Huang, and Cheng-Lin Liu

National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences, Beijing 100090, China.
{ymzhang, kzhuang, liucl}@nlpr.ia.ac.cn

***Abstract*—In this paper, we propose an efficient and robust algorithm for graph-based transductive classification. After approximating a graph with a spanning tree, we develop a linear-time algorithm to label the tree such that the cut size of the tree is minimized. This significantly improves typical graph-based methods, which either have a cubic time complexity (for a dense graph) or $O(kn^2)$ (for a sparse graph with k denoting the node degree). Furthermore, our method shows great robustness to the graph construction both theoretically and empirically; this overcomes another big problem of traditional graph-based methods. In addition to its good scalability and robustness, the proposed algorithm demonstrates high accuracy. In particular, on a graph with $400,000$ nodes (in which $10,000$ nodes are labeled) and $10,455,545$ edges, our algorithm achieves the highest accuracy of 99.6% but takes less than 10 seconds to label all the unlabeled data.**

***Keywords*-graph-based semi-supervised learning; transductive learning; graph mining;**

I. Introduction

In many machine learning applications, labeled data are scarce because labeling data is both time-consuming and expensive. However, unlabeled data are very easy to collect in many applications such as text categorization and image classification. This has motivated machine learning researchers to develop learning methods that can exploit both labeled and unlabeled data during model learning. Such a learning paradigm developed over the past decade is referred to as semi-supervised learning [7].

In this paper, we consider a particular setting of semi-supervised learning called transductive learning in which the unlabeled test data are also available before model training. 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 $\mathbf{x}_i$, $1 \leq i \leq l+u$, is the input of a data point and $y_i \in \{1, \ldots, K\}$, $1 \leq i \leq l$, indicates the class of the labeled data point x_i. The goal is to predict the labels of unlabeled data by utilizing the information from both the labeled and unlabeled data.

Among the most popular transductive learning methods are graph-based methods [25, 24, 1]. Graph-based methods are appealing because: (1) graph provides a powerful tool to describe the similarity between data; (2) for many important applications, like social network, genomic data, web pages etc., problems directly present themselves as graphs.

Despite of these advantages, graph-based methods suffer from two main drawbacks[18]. (1) They are not scalable. Typical graph-based methods usually have a cubic time complexity because of the calculation of the inverse of graph Laplacian. Although the complexity can be reduced to $O(kn^2)$ (k is the node degree) when the graph is sparse, it is still not applicable to large-scale problems. (2) Their performance is very sensitive to the graph construction. For the most popular kNN graph or ϵ-graph, a small change in k or ϵ would make a big difference in accuracy [21, 14]. Thus, one has to carefully tune these parameters in practice, which is however very difficult because of the scarcity of labeled data.

To overcome these problems, in this paper, motivated by the fact that most real-world graphs are sparse, we propose to firstly approximate the graph with a spanning tree, and then label the tree in a way such that the overall cut size is minimized. For a given spanning tree, our labeling algorithm has a linear time and space complexity with respect to the data size which makes our method a perfect choice for large scale problems.

We use a minimum spanning tree (MST) to approximate a graph. In addition to its simplicity, MST is very robust to the graph. For a connected ϵ-graph with the RBF weighting function, we can formally prove that the structure of MST is invariant to the increase of ϵ. 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. Although there are some works that use a tree to approximate a graph, their motivation is to design fast semi-supervised learning algorithm. To our best knowledge, this is the first work to extensively explore the robustness property of tree-based semi-supervised learning methods.

We have performed detailed experiments to confirm the advantage of the proposed method in both scalability and robustness. In term of accuracy, empirical results shows that our method gives good approximations to traditional graph-based methods, while is consistently better than other scalable graph-based methods.

The rest of this paper is organized as follows. In the next section, we introduce the related work. In Section III, we then present our major work including the notations, the

model definition, the detailed algorithm, and the relationship with a previous model. After that, in Section IV, we introduce how to perform transductive classification using our proposed method. In Section V, we provide experimental results to validate the advantages of our method. Finally, we set out the conclusion in Section VI.

II. Related Work

Recently, Herbster et al. [12] 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 was performed on this Laplacian kernel matrix to predict all unlabeled data. By calculating one column of the Laplacian kernel for each trial, their algorithm takes $O(ln)$ time and $O(n)$ space, where n is the number of the data points.

Herbster and Lever [11] considered the online graph prediction problem. They proposed to approximate a general graph with a path graph, and use the nearest neighbor classifier to predict on the path graph. They gave cumulative mistake bounds of their algorithms which show improvements over Laplacian-based methods when the graph has a large diameter. Cesa-Bianchi et al. [6] extended the work of [11] from the un-weighted graph to the weighted one, and shown the cumulative errors of their algorithm is bounded in terms of the cut size of a random spanning tree. The cost of their algorithm is $O(n)$ in both time and space requirement. However, these two methods approximated the original graph by exploiting a path graph, which is basically a naive tree structure, i.e., a line, and hence too simple and inflexible to generate good performance for practical problems. Experimental results later presented in Section V also validate this point.

Cesa-Bianchi et al. [5] characterized the number of mistakes necessary and sufficient for sequentially predicting a given tree, and built an algorithm to achieve this number by minimizing the cut size of the tree. Their method uses the space linear in n, and the running time is $\min\{K, n_f\}K + n \log D_T$ for online learning where K is the cut size of the labeled tree T, D_T is the diameter of T, n_f is the number of nodes in T with the degree bigger than 2. However, for transductive learning, the running time is $O(n^2)$. We will discuss the difference between this work and our method in more details in the next section.

Another family of scalable graph-based semi-supervised learning methods are based on subsampling a small subset from the whole data set. 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. [10] built the matrix by directly setting $w_{ij} = 0$ if neither i nor j is a prototype. Zhang et al. [23] approximated the adjacency matrix by the Nyström method, and Liu et al. [16] approximated it by a two-step transition probability matrix. The running time of all these prototype-based methods are $O(m^2 n)$, 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.

For sparse graph, Spielman and Teng [20] presented a randomized algorithm to solve diagonally dominant linear systems. It can be used to solve Gaussian random fields [25]'s optimization problem in $n \log^{O(1)} n \log \frac{1}{\epsilon}$ time.

III. Predicting a Tree: Minimum Tree Cut Algorithm

In this section, we present the main work in this paper. We first present notations used throughout the paper and then introduce our work in details.

A. Notations

We use G to denote a graph, T to denote a tree, while $E(G)$ and $E(T)$ denote the edge set of graph G and tree T respectively. 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, where K is the number of classes, 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. Objective Function

Blum et al. [3, 4] first proposed to predict the graph with the labeling which minimizes the cut size of the graph, and meanwhile is consistent with all the labeled data. However, for multi-class discrete variables, this optimization problem is NP-hard. Thus, Zhu et al. [25] binarized the problem via a standard one-vs-all scheme, and solved it in the continuous space. Their method is still too expensive to solve large scale problems. Inspired by [12] and [5], we first approximate the graph by a spanning tree, then solve the cut size minimization problem directly on the discrete label set. Formally, the objective function of our method can be formulated as follows:

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

where $E(T)$ is the edge set of the tree T, K is the number of classes, and w_{ij} describes the weight for edge (i, j). As we will show in the next subsection, 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 by [3, 15], 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

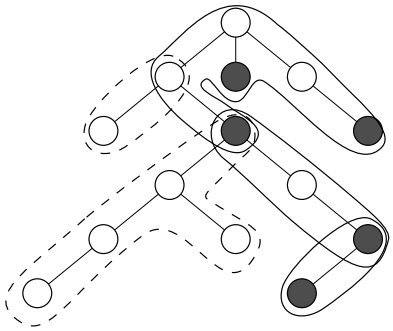


Figure 1. An illustration of lb-tree and sc-tree. In this partially labeled tree, black nodes denote labeled data, and blank nodes are unlabeled. The subtrees in solid lines are lb-trees, and the subtrees in dash lines are sc-trees.

$w_{ij} = \exp\{-|x_i - x_j|^2/\sigma^2\}$. When weights of edges are different, it is very unlikely to have multiple solutions. This phenomenon has actually been observed in our experiments.

C. Minimum Tree Cut Algorithm (MTC)

Before detailing the algorithm, we first define some concepts which will be used extensively in our method. The first one, called *label-bordered tree* (lb-tree), is firstly introduced in [5]. 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. See Figure 1 for an example.

Our method proceeds as follows. First, we split a partially labeled tree T into lb-subtrees and sc-subtrees. Then, we use the proposed MTC algorithm to label each lb-tree. Finally, we label all the nodes in one sc-tree with its connecting node's label, which motivates the name of the sc-tree.

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

In the following, we design a novel method 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 our minimum tree cut

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

Algorithm 1: Tree splitting algorithm

(MTC) algorithm 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

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

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

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

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. Since computing $cutsize(i,k)$ needs $|\downarrow(i)| \times K$ operations, filling the whole table needs $\sum_i \sum_k |\downarrow(i)| \times K = |\downarrow^*(r)| \times K^2$ operations. Therefore, the algorithm needs $O(Kn)$ space and $O(K^2 n)$ operations to compute all lb-trees's $cutsize$ values. When K is big, we could use the one-vs-all scheme to binarize the problem. As solving each binary classification problem needs $O(n)$ operations, the total computational cost is reduced to $O(Kn)$.

Once the $cutsize$ values of the root r are determined, by definition $\min\{cutsize(r,k) : k = 1,\ldots,K\}$ 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

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

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

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

where $\tilde{k}(i) = \arg\min_k \{cutsize(i,k) : k \neq k^*(\uparrow(i))\}$. 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 method exactly solves the optimization problem (1).

Theorem 1. *The MTC method introduced in the subsection III-C optimizes the problem (1) exactly.*

Proof: We denote the optimal value of the problem (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,

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

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 MTC method 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 $\sum_{T_i \in lb-tree(T)} mincut(T_i, \mathbf{y}_i)$. This completes our proof. ■

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

D. Insensitiveness to Graph Construction

In this section, we prove the property that the structure of MST is insensitive to the ϵ-graph theoretically. Hence, the proposed algorithm is very robust to the parameter in graph construction, making it very desirable for practical problems.

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

Proof: To prove this proposition, we first recall Kruskal's minimum spanning tree algorithm which proceeds as follow. 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) $E(G) \subseteq E(G')$; (2) for $\forall e \in E(G)$ and $\forall e' \in E(G') \setminus E(G)$, $\pi_e < \pi_{e'}$. Thus, edges which 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 which implies $E(T) = E(T')$. ■

When the graph is constructed by kNN, we observe the same property in experiments, although we cannot formally prove it. This property makes MST very robust to the graph hyperparameter, and in turn makes the performance of our method very robust to the graph hyperparameter. Experimental results in Section V will further validate this desirable advantage.

E. Relationship with Cesa-Bianchi et al. [5]'s Method

The idea of minimizing the cut size of a partially labeled tree was first used by Cesa-Bianchi et al. [5] in the online graph prediction setting. Roughly speaking, to predict an unlabeled node i in some lb-tree, their method proceeds by performing a post order traversal of the lb-tree from node i. When backtracking to node j after all the children of j have been visited, j is assigned a temporary label given by the majority vote among the temporary or labels of its children. This method can merely guarantee the node i's predicted label minimizes the cut size of T, while the other temporary labels of unlabeled data do not necessarily minimize the cut size. Therefore, to correctly predict each unlabeled node, a traversal of the lb-tree from each unlabeled node should be made; this leads to the running time as $O(n^2)$.

Instead of propagating the temporary labels from bottom to top, our method propagates the minimum cut size which

enables us to label the whole lb-tree effectively in just two traversals.

In addition, our method has the important advantage of being able to handle weighted graphs with more than two classes, while Cesa-Bianchi et al. [5]'s method can only handle un-weighted graphs with two classes.

IV. Transductive Classification with MTC

To apply the MTC algorithm 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 followed by running MTC on T. In case that the graph is not connected, we can run the MTC algorithm on each connected component respectively. As suggested by [12], to further boost the performance, one can use ensembles of trees. Specifically, one can generate many spanning trees, run MTC on each tree, then use the majority vote method to predict unlabeled nodes. In the following, we will discuss different methods for graph construction and spanning tree generation.

A. Tree Construction Methods

Minimum Spanning Tree (MST). MST is a spanning tree that minimizes the total cost. In our setting, the cost of an edge (i, j) is defined as $\pi_{ij} = \frac{1}{w_{ij}}$, thus, MST of G is the spanning tree that solves the problem

$$\min\{\sum_{(i,j)\in E(T)} \pi_{ij} : T \in \mathcal{T}(G)\}, \tag{4}$$

where $\mathcal{T}(G)$ is the set of spanning trees of graph G. According to [12], this problem is equivalent to

$$\max\{\sum_{(i,j)\in E(T)} w_{ij} : T \in \mathcal{T}(G)\}. \tag{5}$$

The above objective function is very intuitive, and formally it says MST best approximates the original graph in term of the trace norm of graph Laplacian. MST can be generated by Kruskal's algorithm in $O(|E| \log n)$ time. For sparse graphs, like kNN graphs, it can be sped up to $O(n \log n)$.

Shortest Path Tree (SPT). SPT is a spanning tree that the distance between a selected node and all other nodes is minimal. Herbster et al. [12] used this tree to approximately solve the problem of,

$$\min\{\text{tr}(T^+ - G^+) : T \in \mathcal{T}(G)\}, \tag{6}$$

where $\text{tr}(A)$ denotes the trace of matrix A, and A^+ denotes the Moore-Penrose pseudoinverse of A. SPT can be generated by Dijkstra's algorithm. With a binary heap, its running time is $O((|E| + n) \log n)$. For sparse graphs, it requires $O(n \log n)$ time.

Random Spanning Tree (RST). RST is a spanning tree taken with the probability proportional to the product of its edge weights. Cesa-Bianchi et al. [6] proposed to use this tree to ensure a worst case bound of their algorithm. Another advantage of RST is that for many graphs, the RST tree can be generated in $O(n)$ time.

B. Graph Construction Methods

Many researchers have shown practically and theoretically that the graph plays a crucial role for the success of graph-based transductive methods [21, 17]. The most popular graph construction methods include kNN graph and ϵ-graph. However, a direct implementation of both types of graphs need $O(dn^2)$ time where d is the data dimension. There are lots of work for efficient graph construction. For example, Chen et al. [8] computed an approximating kNN graph via the divide and conquer method. Plaku and Kavraki [19] developed parallel algorithms to construct graph. Other techniques to speed up the graph construction include the KD-tree [2] and locality sensitive hashing [13]. However, compared to the complexity of spanning tree generation and the MTC algorithm, graph construction still dominates the time of graph-based transductive classification. Note that many methods have been recently proposed to construct graph to better describe the data relationship [14, 9, 22]. However, the high computational cost of these methods prevents them from large scale applications.

V. Experiments

In this section, we evaluate our method by performing extensive experiments on real-world data sets. We compare our method with two tree-based transductive methods: the graph perceptron algorithm (GPA) [12] and the weighted tree algorithm (WTA) [6], and one classical full graph-based method, Gaussian Random Fields (GRF) [25]. We also use 1NN and SVM as baseline methods.

For all graph-based methods, following many works in the literature, we exploit the kNN method to construct graph and the RBF function $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_0^2\}$ to weight the edges, where σ_0^2 is set to $\sum_{(i,j)\in E} \|\mathbf{x}_i - \mathbf{x}_j\|^2/|E|$. For tree-based methods, we fix k to 100, because they are insensitive to this parameter (as validated theoretically in Section III-D and empirically in the next subsection). For GRF, we manually choose k such that the test error is minimized.

As reported by [12] and [6], the performance of MST is always the best among different tree construction methods. We also observed similar results in our experiments. Hence, we only report the classification results of different methods on MST.

A. Small Data Sets

In this subsection, to show the effectiveness of the proposed MTC algorithm, we run experiments on two small size data sets[1]: (1) COIL20 data set. This is an image data set which contains 20 objects. The images of each object were taken 5 degrees apart and each object has 72 images. The

[1] http://www.zjucadcg.cn/dengcai/Data/MLData.html

size of each image is 32×32 pixels, with 256 grey levels per pixel; (2) USPS data set. It contains $7,291 + 2,007 = 9,298$ handwritten digit images of size 16×16. Note that, for small data sets, all the graph based methods can perform the learning very efficiently. We hence focus on examining the classification accuracy of different methods here and leave the efficiency evaluations on two large-scale data sets in Section V-B.

1) Classification Accuracy: For each data set, we randomly select l data points as labeled data such that they contain at least one sample from each class, and leave all the other data as unlabeled. Then we perform various learning methods on this partially labeled data set, and record the classification accuracy on the unlabeled data. To control the variance of results, we repeat the procedure 10 times for different training/testing splits, and report the average classification accuracy in Figure 2.

As observed from the results, the following conclusions can be made. (1) Among all methods of approximating graph by tree, our proposed MTC method achieves the best classification accuracy in all cases. (2) MTC gives a very good approximation to the full graph-based method GRF, and is even better than GRF on the COIL20 data set. This can be explained by the fact that real-world graphs are usually sparse. (3) Transductive methods overwhelm the supervised methods (i.e., the RBF-SVM and 1NN) when the number of labeled data is limited, while this effect becomes less obvious as the number increases.

2) Robustness to Graph Construction: One major problem of graph-based transductive learning is that their performance is very sensitive to the graph construction. For the kNN graph or ϵ-graph, a small perturbation in k or ϵ and σ^2 will result in big changes in accuracy. In this section, we validate the theory that using the minimum spanning tree to approximate a graph can solve this problem effectively.

In this experiment, we fix $l = 200$ for COIL20 and $l = 400$ for USPS and construct kNN graphs or ϵ-graphs for different k or ϵ and σ^2. We then perform GRF and MTC on the resulting graphs. Specifically, for kNN graphs, we vary k in the set $\{2^1, 2^2, 2^3, 2^4, 2^5, 2^6\}$ with $\sigma^2 = \sigma_0^2$, and vary σ^2 in the set $\sigma_0^2 * \{2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3\}$ with $k = 4$. Similarly, for ϵ-graphs, we draw the accuracy curves respectively by varying ϵ with a fixed σ. The curves of varying σ with a fixed ϵ demonstrated very similar trend with those of kNN graphs and we hence do not draw them for succinctness.

Figure 3 shows the average classification accuracy over 10 random splits. As we can see, the performance of MTC is almost invariant to the variance of k, σ^2, ϵ. In fact, this effect is due to the robustness of MST with respect to variance of k, ϵ, and σ^2, which we have analyzed in Section III-D. This property helps MST to filter out noisy edges, which however may severely hurt the performance of GRF.

B. Large Data Sets

We conduct a series of experiments on two large scale data sets in this section. We will first report results on the MNIST data[2] and then a Web-spam Data Set[3]. All methods are performed on an HP server with a 2.27 GHz Xeon 4 Core CPU, and 16 GB RAM.

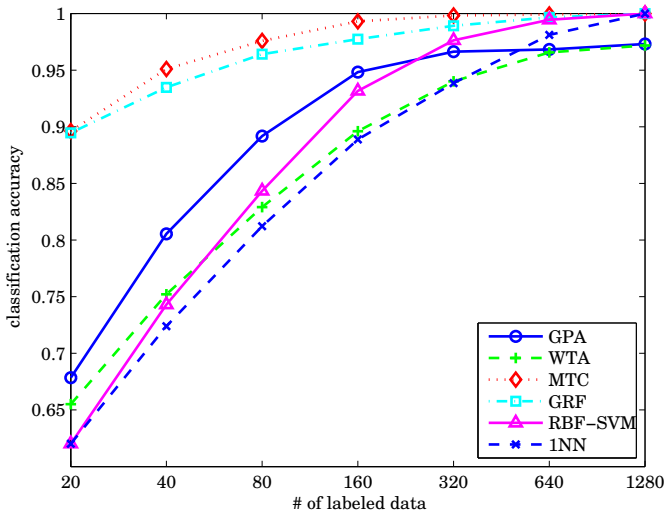
1) MNIST Data Set: The MNIST data set contains $60,000 + 10,000 = 70,000$ handwritten digit images. The size of each image is 28×28 pixels, with 256 grey levels per pixel. Our experiment proceeds as follows. First, we randomly select n data points from $70,000$ images, and construct graph on these data. Then 400 nodes are randomly selected as labeled data, and different algorithms are performed to predict unlabeled nodes. To observe the effect of unlabeled data, we vary n from $10,000$ to $70,000$, and report the average classification accuracy over 10 times training/testing splits and running time in Figure 4. The reported time includes time for generating MST and labeling the spanning tree. The time for graph construction is not included.

Several observations are highlighted as follows. (1) Similar to the results in COIL and USPS, MTC gives a good approximation to GRF, and achieves the best accuracy among all the tree-based methods. (2) Unlabeled data indeed help the classification. In general, as the number of unlabeled data increases, the accuracy increases correspondingly. (3) While the performance of MTC and WTA improves consistently, the accuracy of GPA decreases when the size of unlabeled data is very huge. This is due to the effect of overfitting. For GPA, we are training a linear classifier in a feature space of $70,000$ dimensions with 400 training samples. This also leads to the same problem. In comparison, the tree-based methods MTC and WTA exploit a much simpler structure and hence demonstrate very robust performance. (4) From Figure 4(b), our proposed linear-time MTC shows very good scalability as the size of training samples increases. The WTA algorithm is also very fast due to its linear complexity. However, as WTA basically adopts a too simple tree, i.e., a path graph, to approximate the graph. This makes its performance usually worse than our proposed MTC method.

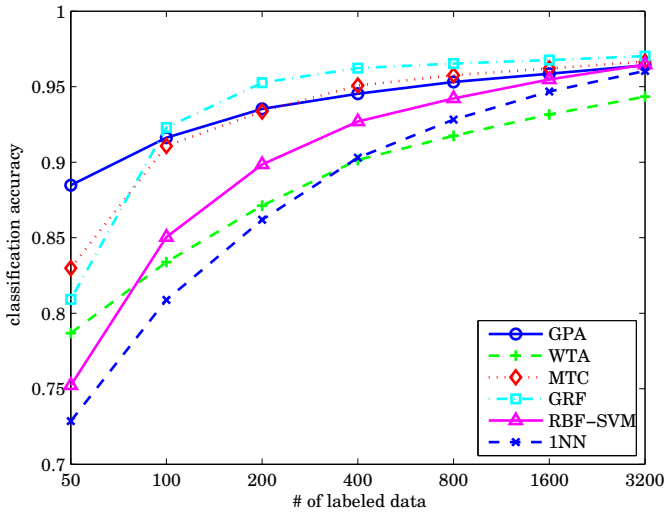
2) Web-spam Data Set: We also apply our method to the 2007 web-spam challenge developed by the University of Paris VI. It contains $400,000$ web pages. A web-page is connected to another web-page if there is at least one hyperlink from the former to the latter. Thus, the links are directed. Following [12], 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.

[2]http://yann.lecun.com/exdb/mnist/

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



(a)



(b)

Figure 2. Average classification accuracy for different labeled data size: (a) COIL20 data set; (b) USPS data set.

There are about 80% of non-spam 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 I. Due to the same experimental setup, the results except MTC and WTA are directly copied from [12]. For ensemble GPA, 21 minimum spanning trees are used, and predictions are made by majority voting. Witshel et al., Filoce et al., Benczur et al. are three methods that participated the 2007 web-spam challenge. We did not report the result of GRF as it is intractable to run GRF on this large data set. GPA takes about 30 minutes to train a single kernel perceptron. In comparison, our proposed MTC takes less than 10 seconds to complete the classification. Similarly, WTA is also very fast and finishes the learning also within 10 seconds. However, the classification accuracy of WTA is 99.0, which is significantly 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.

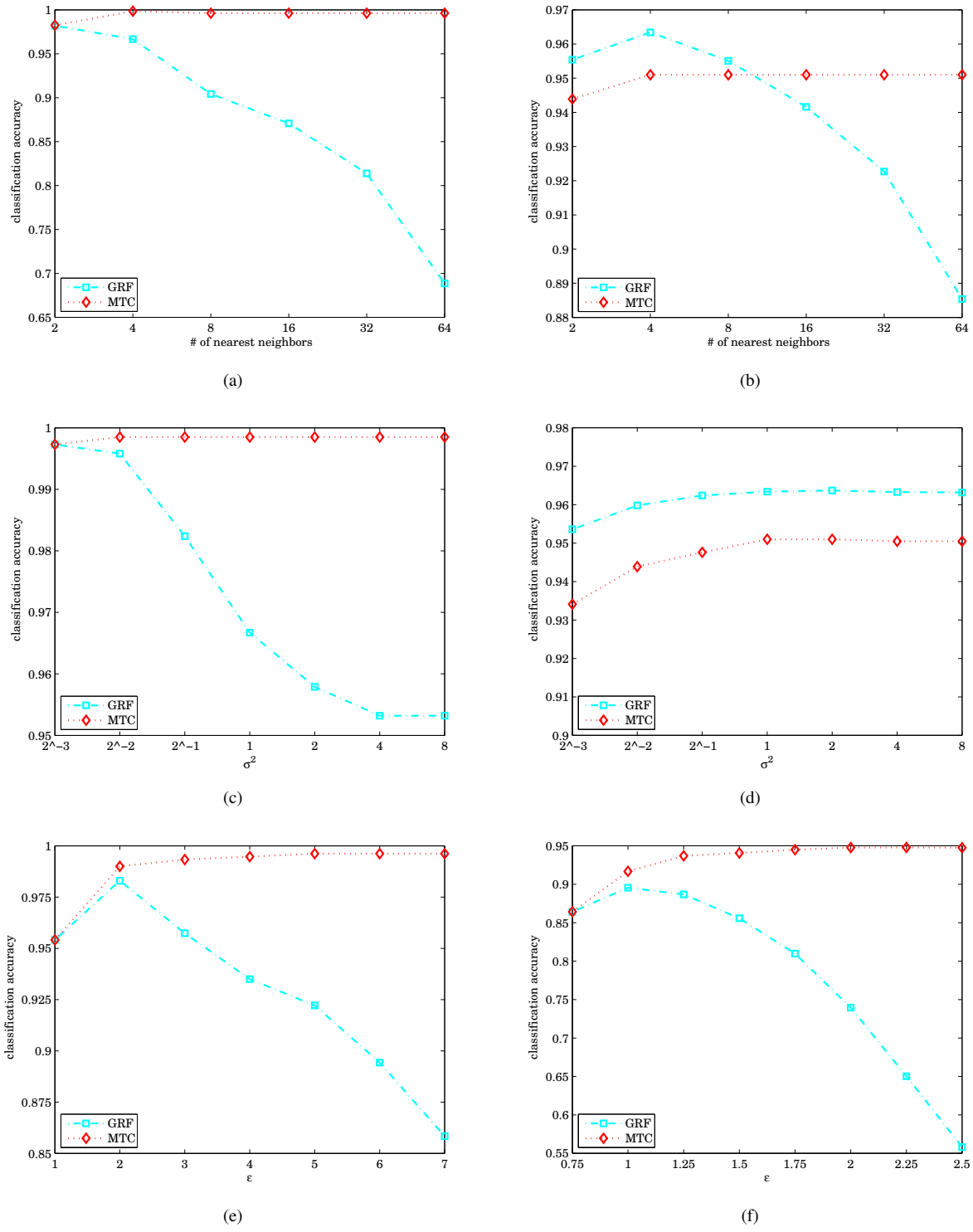
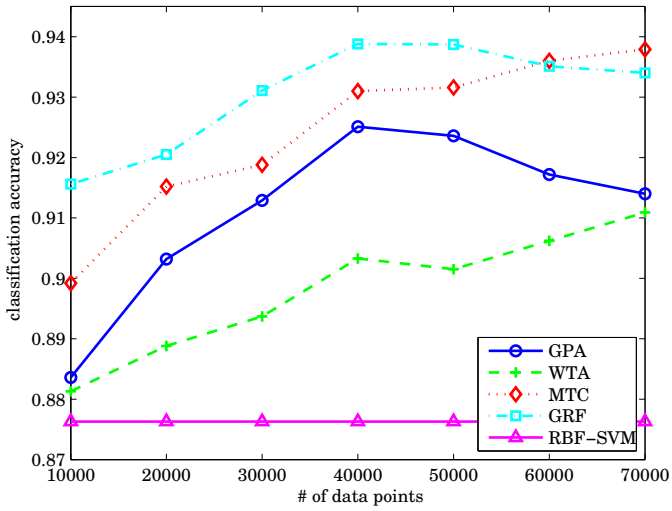
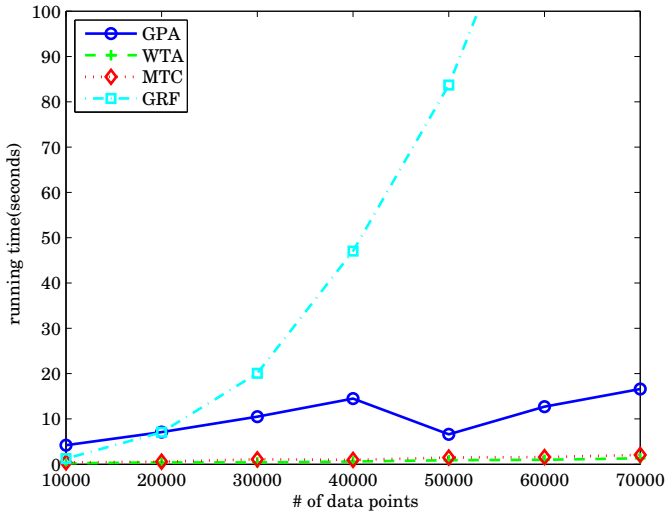


Figure 3. Robustness of GRF and MTC to kNN graphs and ϵ-graph: (a),(b) average classification accuracy for different k on COIL and USPS; (c),(d) average classification accuracy for different σ^2 on COIL and USPS; (e),(f) average classification accuracy for different ϵ on COIL and USPS.



(a)



(b)

Figure 4. Experimental results on MNIST for different data size: (a) average classification accuracy; (b) running time.

VI. Conclusion

In this paper, we proposed a fast graph-based transductive classification method. Inspired by the sparsity of the real world graphs, we first exploited a spanning tree to approximate the graph. Based on minimization of the cut size, we then developed a highly robust and efficient learning algorithm to label the tree. Our algorithm has a linear complexity in terms of both time and space. This is significantly distinctive with typical graph-based methods, which either have a cubic time complexity (for a dense graph) or a quadratic complexity (for a sparse graph). We evaluated our proposed method on four real world data sets (including two large-scale sets). Experimental results showed that our proposed method can usually be superior to other competitive approaches in terms of both classification accuracy and learning efficiency. Future work includes the parallel application of multiple MTC's for different lb-trees.

Acknowledgment

The work was supported by the National Natural Science Foundation of China (NSFC) under grants No. 60825301 and No. 61075052. The authors want to thank Sergio Ro-

Table I
CLASSIFICATION ACCURACY ON WEB-SPAM DATA SET.

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

jas Galeano, Fabio Vitale and Giovanni Zappella for their helpful discussion.

REFERENCES

[1] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[2] J.L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.

[3] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the International Conference on Machine Learning*, 2001.

[4] A. Blum, J. Lafferty, M.R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the International Conference on Machine Learning*, 2004.

[5] N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction of a labeled tree. In *The 22th Annual Conference on Learning Theory*, 2009.

[6] N. Cesa-Bianchi, C. Gentile, I.F. Vitale, and G. Zappella. Random spanning trees and the prediction of weighted graphs. In *Proceedings of the International Conference on Machine Learning*, 2010.

[7] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[8] J. Chen, H. Fang, and Y. Saad. Fast approximate k-NN graph construction for high dimensional data via recursive Lanczos bisection. *The Journal of Machine Learning Research*, 10:1989–2012, 2009.

[9] S.I. Daitch, J.A. Kelner, and D.A. Spielman. Fitting a graph to vector data. In *Proceedings of the International Conference on Machine Learning*, 2009.

[10] O. Delalleau, Y. Bengio, and N. Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[11] M. Herbster and M. Lever, G. andPontil. Online prediction on large diameter graph. In *Advances in Neural Information Processing Systems*, 2008.

[12] M. Herbster, M. Pontil, and S. R. Galeano. Fast predciton on a tree. In *Advances in Neural Information Processing Systems*, 2008.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[14] T. Jebara, J. Wang, and S.F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, 2009.

[15] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education, 2006.

[16] W. Liu, J.F. He, and S.F. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, 2010.

[17] M. Maier, U. Von Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems*, 2009.

[18] G.S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the International Conference on Machine Learning*, 2007.

[19] E. Plaku and L.E. Kavraki. Distributed computation of the knn graph for large high-dimensional point sets. *Journal of Parallel and Distributed Computing*, 67(3):346–359, 2007.

[20] D.A. Spielman and S.H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *Arxiv preprint cs/0607105*, 2006.

[21] F. Wang and C. Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008.

[22] S. Yan and H. Wang. Semi-supervised learning by sparse representation. In *SIAM International Conference on Data Mining, SDM*, pages 792–801, 2009.

[23] K. Zhang, J.T. Kwok, and B. Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, 2009.

[24] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2004.

[25] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.