FLAG: Faster Learning on Anchor Graph with Label Predictor Optimization

Weijie Fu, Meng Wang, *Senior Member, IEEE,* Shijie Hao, and Tingting Mu, *Member, IEEE*

Abstract—Knowledge graphs have received intensive research interests. When the labels of most nodes or datapoints are missing, anchor graph and hierarchical anchor graph models can be employed. With an anchor graph or hierarchical anchor graph, we only need to optimize the labels of the coarsest anchors, and the labels of datapoints can be inferred from these anchors in a coarse-to-fine manner. The complexity of optimization is therefore reduced to a cubic cost with respect to the number of the coarsest anchors. However, to obtain a high accuracy when a data distribution is complex, the scale of this anchor set still needs to be large, which thus inevitably incurs an expensive computational burden. As such, a challenge in scaling up these models is how to efficiently estimate the labels of these anchors while keeping classification performance. To address this problem, we propose a novel approach that adds an anchor label predictor in the conventional anchor graph and hierarchical anchor graph models. In the proposed approach, the labels of the coarsest anchors are not directly optimized, and instead, we learn a label predictor which estimates the labels of these anchors with their spectral representations. The predictor is optimized with a regularization on all datapoints based on a hierarchical anchor graph, and we show that its solution only involves the inversion of a small-size matrix. Built upon the anchor hierarchy, we design a sparse intra-layer adjacency matrix over these anchors, which can simultaneously accelerate spectral embedding and enhance effectiveness. Our approach is named Faster Learning on Anchor Graph (FLAG) as it improves conventional anchor-graph-based methods in terms of efficiency. Experiments on a variety of publicly available datasets with sizes varying from thousands to millions of samples demonstrate the effectiveness of our approach.

Index Terms—Semi-supervised learning, graph-based learning, machine learning

1 Introduction

Knowledge graphs, which organize information in a structured way by describing the relationships among entities, have received much attention from both academia and industry in the past few years [16], [17], [36]. Well-known knowledge graph systems include Google Knowledge Graph, Probase, DBPedia, YAGO, and TrueKnowledge. In a knowledge graph, entities are denoted as nodes or datapoints, categories are their labels, and relationships are directed links between these datapoints [32]. However, in most data mining applications, the labels of many datapoints are missing, and it is often prohibitively labor-intensive and time-consuming to collect their labels. In contrast, the amount of unlabeled data can be huge in many domains. Semi-Supervised Learning (SSL), which exploits the prior knowledge from both unlabeled and labeled data, thus attracts considerable attention to estimate the missing labels. In recent years, various SSL methods have been developed, such as mixture methods [6], co-training [4], [48], semi-supervised support vector machines [7], [18], and graph-based methods [1], [35], [53].

In this paper, we focus on Graph-based SSL (GSSL), which is one of the most successful SSL approaches and shows the state-of-art performance in many areas [29], [45], [54]. In general, GSSL first constructs an adjacency graph to capture the data distribution for all datapoints, where the weight of the edge between two nodes represents their similarity. As a consequence, the labels for classification can be propagated from limited labeled data to remaining unlabeled data on the graph. This intuitive interpretation of the label propagation also offers GSSL more expansibility and many novel graph models have been recently proposed, such as hypergraphs for modeling higher-order relevances [51], and multigraphs for integrating multi-view features [43].

Despite the success in a variety of applications, the traditional GSSL approaches have a bottleneck in dealing with large-scale data. They face a quadratic complexity in graph construction. In addition, denoting N as the number of datapoints, the optimal solution of GSSL requires the inversion of a graph Laplacain matrix with the size $N \times N$ [53], which requires a computational cost of $O(N^3)$. The costs thus become impractical for large-scale datasets.

Recent works seek to employ anchors to build fast graph-based learning methods [23], [42], [47]. Here anchors refer to the points that roughly cover the data distribution in a feature space[1]. Specifically, these methods first estimate the inter-layer adjacency weights between datapoints and anchors, and then infer the labels of datapoints from these anchors. Consequently, they reduce the scale of the matrix inversion to the size of the anchor set and correspondingly speed up the optimization. However, to guarantee classification accuracies, anchors in these approaches need to be dense when the data distribution is complex, which leads to

- *W. Fu, M. Wang, S. Hao are with the School of Computer and Information, Hefei University of Technology, 230009, China (E-mail: {fwj.edu, eric.mengwang, hfut.hsj}@gmail.com).*
- *T. Mu is with the School of Computer Science, University of Manchester, Manchester M13 9PL, U.K. (e-mail: tingting.mu@manchester.ac.uk).*

[1]Throughout the manuscript, we call the space of the raw data representation as "feature space", and the one spanned by the eigenvectors of Graph Laplacian as "spectral space" [31].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

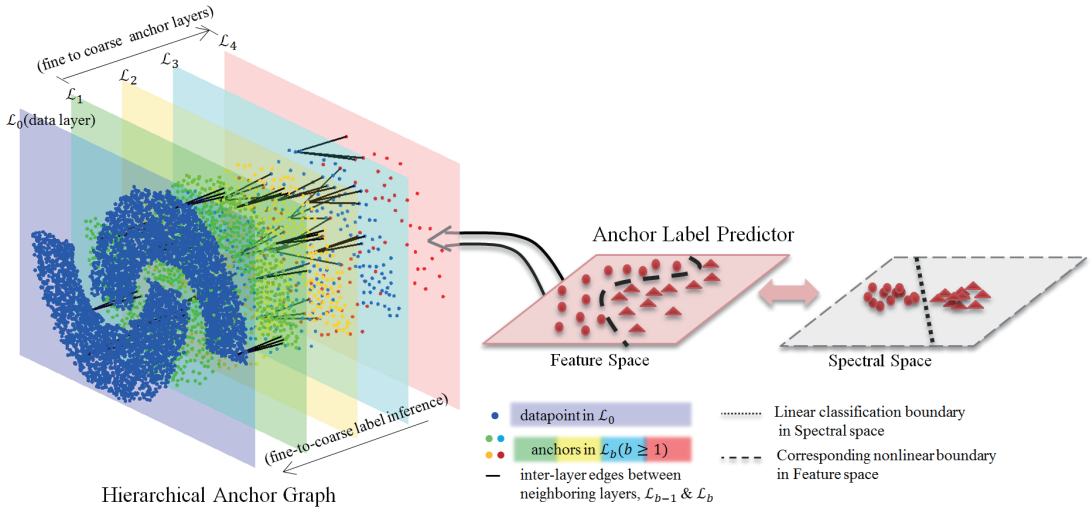


Fig. 1. An illustrative example of FLAG, where the labels of the coarsest anchors are first obtained based on their spectral representations with an anchor label predictor, and then spread to datapoints in a coarse-to-fine manner with the inter-layer adjacency matrices. Note that owing to the nonlinear spectral embedding, the linear predictor in FLAG actually corresponds to a nonlinear classification boundary in the feature space.

huge computational costs for large-scale datasets. Recently, a hierarchical anchor graph model with a pyramid-style structure is proposed in [41], which explores multiple-layer anchors to perform hierarchical label inference layer by layer. As a result, this approach can enlarge its finest anchor layer to improve classification performance. However, since the labels of all datapoints are essentially inferred from the coarsest anchors, to obtain a high accuracy, the number of these anchors still needs to be large. Like the pervious approaches, the computational complexity for optimizing their labels can be increasingly expensive. Therefore, these fast anchor-based methods are still insufficiently efficient to handle large-scale datasets with complex data distributions.

To address this issue, in this paper we develop a novel approach called Faster Learning on Anchor Graph (FLAG), which further scales up anchor-graph-based methods. As illustrated in Fig.1, instead of directly optimizing the labels of the coarsest anchors in hierarchical anchor graph models[2], we estimate their labels with a predictor and optimize the predictor in a hierarchical-anchor-graph-based regularization framework. To keep computational efficiency, we need to employ a linear predictor with a small hypothesis space. Therefore, we first design a sparse intra-layer adjacency matrix over the coarsest anchors with anchor hierarchy. Then we perform spectral embedding on these anchors and build their low-dimensional but discriminative spectral representations. In this way, the optimization can be computed with the matrix inversion, where the matrix size is just equivalent to the dimensionality of the spectral representation.

The rest of this paper is organized as follows. In Section 2, we briefly review related work on graph-based learning algorithms. In Section 3, we introduce the formulation of the anchor-based learning framework and analyze its dilemma. We propose our faster learning approach in Section 4. In Section 5, we make comparisons with other approaches. We finally conclude this paper in Section 6.

[2]Note that we may not further discriminate anchor graphs and hierarchical anchor graphs, as anchor graph is actually just a case of hierarchical anchor graph with an individual anchor layer.

2 Related Work

In this section, we focus on the related work on improving the efficiency of GSSL from two aspects, namely, graph construction and model optimization.

To reduce the time cost in the first aspect, many fast graph construction approaches have been developed. Chen et al. [8] first proposed to construct an approximate kNN graph for high-dimensional data via recursive Lanczos bisection. To deal with arbitrary similarity measures, Dong et al. [10] subsequently presented a simple but efficient solution based on local search algorithms. In [39], Wang et al. also proposed a multiple random divide-and-conquer approach for the graph construction and additionally presented a neighborhood propagation scheme to improve its effectiveness. To deal with the data distributed over commodity clusters, Goyal et al. [13] proposed a distributed online approaches based on sketching algorithms, and introduced it into natural language processing. Besides, Wang et al. [40] employed parallel auction algorithms to recover a sparse yet nearly balanced subgraph for social networks, which significantly reduces computational costs. As a powerful method, Approximate Nearest Neighbor Searching (ANNS) is also used to build big graphs. For example, Zhang et al. [49] employed Locality-Sensitive Hashing into the graph construction. Wang et al. [41] introduced a tree-based ANNS algorithm [30] to accelerate their weight estimation.

In spite of the progress in the fast graph construction, most GSSL methods remain challenging due to their cubic computational complexity in the optimization. To address this issue, Tsang et. al [37] introduced an ϵ-insensitive constraint into traditional LapSVM problems [27]. As a result, its optimization turns to a minimum enclosing ball problem and can be solved with core vector machines [38]. However, to obtain a satisfying accuracy, ϵ needs to be small, which makes this method close to the original LapSVM with a

TABLE 1
Notations and definitions

Notation	Definition
$\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{Z}\}$	A hierarchical anchor graph model, where $\mathcal{X}$ and $\mathcal{U}$ indicate the sets of datapoints and anchors, respectively, and $\mathcal{Z}$ indicates the set of inter-layer adjacency edges between different sets of points.
N_0	The number of datapoints.
C	The number of classes.
l	The number of labeled datapoints.
h	The number of anchor layers.
$\mathcal{L}_b$	The bth layer in the pyramidal graph structure, where $\mathcal{L}_0$ is the layer of raw data and $\mathcal{L}_b (b \geq 1)$ denotes the b-th anchor layer.
N_b	The number of anchors in $\mathcal{L}_b (b \geq 1)$.
$\mathbf{Z}^{a,b}$	The inter-layer adjacency matrix between $\mathcal{L}_a$ and $\mathcal{L}_b$.
$Z_{is}^{a,b}$	The inter-layer adjacency weight between point i in $\mathcal{L}_a$ and point s in $\mathcal{L}_b$.
$\mathbf{W}$	The intra-layer adjacency matrix over all datapoints.
$\mathbf{\Lambda}$	The diagonal matrix of the degrees of the finest anchors.
$\mathbf{A}$	The soft label matrix of the coarsest anchor set.
$\mathbf{F}$	The soft label matrix of the datapoint set.
$\mathbf{Y}_{\mathrm{L}}$	The class indicator matrix on labeled datapoints.
$\tilde{\mathbf{L}}$	The reduced Laplacian matrix in regularization.
$\mathbf{Z}^{\mathrm{H}}$	The cascaded inter-layer adjacency matrix of a hierarchical anchor graph.
$\tilde{\mathbf{P}}$	The linear label predictor on the spectral representation.
$\tilde{\mathbf{W}}$	The intra-layer adjacency matrix over the coarsest anchors.
$\mathbf{\Sigma}$	The diagonal matrix of the degrees of the coarsest anchors.
$\tilde{\mathbf{U}}$	The the spectral representations of the coarsest anchors.

huge cost. Chen et al. [9] presented a method to combine an original kernel with the adjacency graph for scalable manifold regularization, whose cost is still larger than square in practice. Meanwhile, since the above algorithms are merely designed for binary classification, they can only learn individual classifiers for different classes. Benefitting from the development of parallel computation platforms, such as Mapreduce, many parallel algorithms are also proposed to accelerate the model optimization [2], [26], [33].

More recent works seek to employ anchors for scaling up graph-based learning, which can reduce the computational costs of both the graph construction and the model optimization. Zhang et al. [46], [47] first suggested employing a set of anchors to perform a low-rank approximation of a data distribution and span an effective model for label reconstruction. However, since it requires a dense weight matrix to build the relationships between each datapoint and all anchors, its storage requirement becomes impractical for large-scale datasets. Liu et al. [23], [24] then presented anchor graph models by constructing the inter-layer edges between datapoints and their nearby anchors. Besides, they also introduced a geometric reconstruction method for their weight estimation to improve its effectiveness. In [42], Wang et al. improved the efficiency of the reconstruction problem by introducing a new constrict, and alternatively proposed to build an intra-layer adjacency matrix over anchors rather than datapoints. Nevertheless, to obtain a reasonable accuracy, a large-size anchor set is required. As a consequence, the computational costs of the geometric reconstruction and the model optimization can be expensive. Wang et al. [41] therefore extended anchor graph models into a pyramid-style structure by exploring multiple anchor sets, which can improve the classification accuracy by introducing a finer anchor set while fixing the size of the coarsest anchor set. As a result, it carries out hierarchical label inference from the coarsest anchors in a coarse-to-fine manner, and its optimization only involves the inversion of a matrix with the size of the coarsest anchor set. Although the size of this coarsest anchor set can be forced to be small for a lower complexity, the classification performance will accordingly become worse. To obtain a high accuracy, the number of these anchors still need to be relatively large, and the time cost can be expensive.

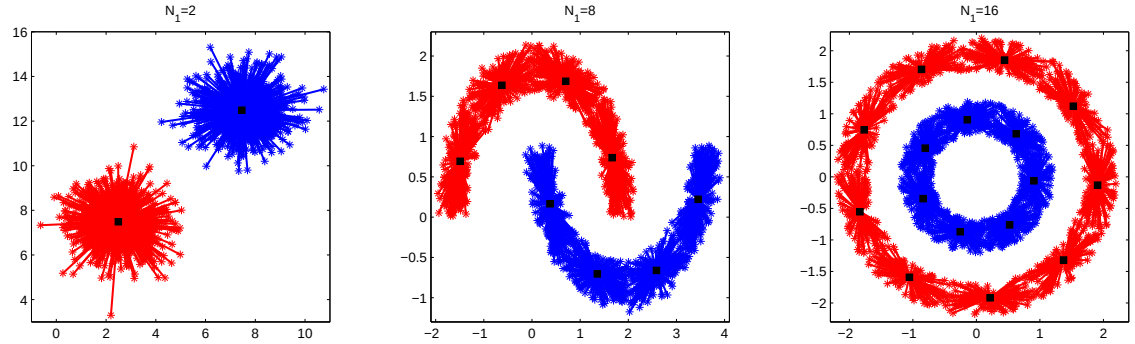
3 Anchor-Graph-Based Learning

In this section, we first review the traditional fast learning approaches built upon anchor graph and hierarchical anchor graph models, and then make an analysis on their dilemma. For convenience, some important notations used throughout the paper and their explanations are listed in Table 1.

Given a dataset $\mathcal{X} = \{\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_{N_0}\} \in \mathbb{R}^{N_0 \times D}$ with the first l samples being labeled from C distinct classes, these approaches aim at classifying the remaining unlabeled data according to their dependence on a hierarchical anchor graph. For this purpose, multiple sets of representative anchors $\mathcal{U}_b \in \mathbb{R}^{N_b \times D} (b = 1, \ldots, h)$ with fine-to-coarse sizes, namely $N_1 > \cdots > N_h$, are first generated. These anchors can be obtained by a clustering process or a random selection [23]. Then suppose the raw datapoint set locates at the bottom layer ($\mathcal{L}_0$) of the pyramid, and the remaining layers ($\mathcal{L}_b, b = 1, \ldots, h$) are all composed of multiple anchor sets. A hierarchical anchor graph can be constructed by linking all these layers up to a pyramid-style structure with the inter-layer adjacency matrices between neighboring layers, represented by $\{\mathbf{Z}^{0,1}, \ldots, \mathbf{Z}^{h-1,h}\}$.

For simplicity, we first consider the inter-layer adjacency matrix between the datapoint set in $\mathcal{L}_0$ and the finest anchor set in $\mathcal{L}_1$, namely $\mathbf{Z}^{0,1} \in \mathbb{R}^{N_0 \times N_1}$. The entries in the i-th row of this matrix are the weights between the i-th datapoint and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(a) (b) (c)

Fig. 2. The toy examples with increasing complexities of data distributions. For each datapoint, we only show its inter-layer adjacency edge with the largest weight in the corresponding anchor graph ($h = 1$). Note that we increase N_1 by $2, 2^2, \ldots,$ until all the edges are "reliable". As we can see, compared with Gaussian data (a), when data distributions become more complex (b-c), more anchors are needed to build the "reliable" inter-layer adjacency edges between datapoints and anchors in the same class.

its nearest anchors in $\mathcal{L}_1$, which can be determined by the kernel regression [12]:

$$Z_{is}^{0,1} = \frac{K_\delta(\mathbf{x}_i, \mathbf{u}_s)}{\sum_{s' \in \langle i \rangle} K_\delta(\mathbf{x}_i, \mathbf{u}_{s'})} \quad \forall s \in \langle i \rangle, \tag{1}$$

where δ is the bandwidth of the Gaussian kernel, and the notation $\langle i \rangle \subseteq [1 : N_1]$ denotes the indices of the k closest anchors of $\mathbf{x}_i$. Then we can conduct the similar kernel regression procedure to estimate the remaining inter-layer adjacency matrices. That is, to obtain $\mathbf{Z}^{b-1,b}$ between two neighboring anchor layers, we have:

$$Z_{is}^{b-1,b} = \frac{K_\delta(\mathbf{u}_i, \mathbf{u}_s)}{\sum_{s' \in \langle i \rangle} K_\delta(\mathbf{u}_i, \mathbf{u}_{s'})} \quad \forall s \in \langle i \rangle, 1 < b \leq h \tag{2}$$

where $\mathbf{u}_i$ is the i-th anchor in $\mathcal{L}_{b-1}$ and $\mathbf{u}_s$ denotes the s-th anchor in $\mathcal{L}_b$. In practice, for large-scale datasets, we can speed up the adjacency matrix estimation with ANNS techniques [30]. As such, the time cost of hierarchical anchor graph construction can be reduced to $O(N_0 \log N_1 D)$.

Let $\mathbf{A} = [\mathbf{a}_1; \mathbf{a}_2; \ldots; \mathbf{a}_{N_h}] \in \mathbb{R}^{N_h \times C}$ denote the optimized label matrix of the coarsest anchor set in $\mathcal{L}_h$. With the anchor hierarchy, the label matrix of the datapoint set $\mathbf{F} \in \mathbb{R}^{N_0 \times C}$ can be inferred from this anchor set in a coarse-to-fine manner:

$$\mathbf{F} = \mathbf{Z}^{0,1}(\ldots(\mathbf{Z}^{h-1,h}\mathbf{A})) = \mathbf{Z}^{\mathrm{H}}\mathbf{A}, \tag{3}$$

where $\mathbf{Z}^{\mathrm{H}}$ denotes the cascaded inter-layer adjacency matrix between the data layer and the coarsest anchor layer. As pointed in [41], this adjacency matrix naturally introduces adaptive inter-layer adjacency relationships between each datapoint and its nearby coarsest anchors.

Meanwhile, an intra-layer adjacency matrix over datapoints, represented by $\mathbf{W}$, can be obtained based on the inter-layer adjacency weights between $\mathcal{L}_0$ and $\mathcal{L}_1$:

$$\mathbf{W} = \mathbf{Z}^{0,1}\mathbf{\Lambda}^{-1}\mathbf{Z}^{0,1\mathrm{T}} \in \mathbb{R}^{N_0 \times N_0}, \tag{4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with $\Lambda_{ss} = \sum_{i=1}^{N_0} Z_{is}^{0,1}$. As we can see that, $W_{ij} > 0$ means two datapoints share at least one finest anchor.

Let $\mathbf{Y}_{\mathrm{L}} = [\mathbf{y}_1; \ldots; \mathbf{y}_l] \in \mathbb{R}^{l \times C}$ denote the class indicator matrix on labeled datapoints, where $y_{ir} = 1$ if $\mathbf{x}_i$ belongs to class r, and $y_{ir} = 0$ otherwise. To deal with a standard multi-class SSL problem, Hierarchical Anchor Graph Regularization (HAGR) [41] is formulated by minimizing the following problem:

$$\begin{aligned} \mathcal{Q}_{\mathbf{A}} &= \sum_{i=1}^{l} \| \mathbf{Z}_{i\cdot}^{\mathrm{H}}\mathbf{A} - \mathbf{y}_i \|^2 + \frac{\lambda}{2} \sum_{i,j=1}^{N_0} W_{ij} \| \mathbf{Z}_{i\cdot}^{\mathrm{H}}\mathbf{A} - \mathbf{Z}_{j\cdot}^{\mathrm{H}}\mathbf{A} \|^2 \\ &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}\mathrm{T}}(\mathbf{I} - \mathbf{W})\mathbf{Z}^{\mathrm{H}}\mathbf{A}) \\ &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\widetilde{\mathbf{L}}\mathbf{A}), \end{aligned} \tag{5}$$

where $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm, λ is the trade-off parameter balancing different terms, $\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}$ is the labeled part of $\mathbf{Z}^{\mathrm{H}}$, and

$$\widetilde{\mathbf{L}} = \mathbf{Z}^{\mathrm{H}\mathrm{T}}\mathbf{Z}^{\mathrm{H}} - \mathbf{Z}^{\mathrm{H}\mathrm{T}}\mathbf{Z}^{0,1}\mathbf{\Lambda}^{-1}\mathbf{Z}^{0,1\mathrm{T}}\mathbf{Z}^{\mathrm{H}} \in \mathbb{R}^{N_h \times N_h}. \tag{6}$$

With the native spare matrix multiplication [44], the cost of Eq.6 scales as $O(N_0 N_h k)$, as all the inter-layer adjacency matrices in $\mathbf{Z}^{\mathrm{H}}$, namely $\mathbf{Z}^{b-1,b}$s, are k-sparse.

Differentiating $\mathcal{Q}_{\mathbf{A}}$ with respect to $\mathbf{A}$ and setting it to zero, we can obtain an optimal solution in the closed-form:

$$\mathbf{A} = (\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}} + \lambda\widetilde{\mathbf{L}})^{-1}\mathbf{Z}_{\mathrm{L}}{}^{\mathrm{T}}\mathbf{Y}_{\mathrm{L}}. \tag{7}$$

Evidently, this matrix inversion takes a cost of $O({N_h}^3)$, namely, a cubic cost with respect to the number of the coarsest anchors.

Note that when the above hierarchical anchor graph model becomes an anchor graph with an individual anchor layer, namely $h = 1$, HAGR degrades to the Anchor Graph Regularization (AGR) method [23].

The above fast anchor-based approaches have shown their promising results for large-scale SSL. However, according to Eq.3, the effectiveness of the label inference highly depends on the distribution of the coarsest anchors. To obtain a high accuracy, the cascaded inter-layer relationships are supposed to be "reliable", namely, connections only exist between the datapoints and the coarsest anchors within the same class. For this purpose, when a data distribution becomes more complex, the required number of the coarsest anchors will increase dramatically. To make it intuitive,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

Figs.2a-2c illustrate this observation by increasing the complexity of the data distribution (for simplicity, we construct anchor graphs with h=1). As we can see, to build "reliable" inter-layer edges, two anchors are sufficient for the example in Fig.2a. However, for the examples in Fig.2b-2c, more anchors are clearly needed. According to Eq.7, it can lead to a large computational burden in the model optimization [41]. In summary, since a large-size coarsest anchor set is required for obtaining a high accuracy, AGR and HAGR still face a dramatic increase of the computational cost for optimizing their labels.

4 Faster Learning on Anchor Graph(FLAG)

To address the above issue, we propose the FLAG model. It employs a label predictor on the spectral representations of the coarsest anchors to estimate their labels, which is optimized in a regularization framework over all datapoints. We first propose the formulation of learning with an anchor label predictor in Section 4.1. We introduce the efficient estimation of spectral representations by employing a sparse intra-layer adjacency matrix over anchors in Section 4.2. The complete FLAG model and its optimization are presented in Section 4.3, followed by a complexity analysis in Section 4.4.

4.1 Learning with Label Predictor Optimization

Instead of directly optimizing the labels of the coarsest anchors, we estimate the labels of these anchors with a label predictor. Let $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{u}_{N_h}] \in \mathbb{R}^{N_h \times D}$ denote the raw representation of the coarsest anchor set. Then based on the anchor label predictor $p : \mathcal{R}^D \rightarrow \mathcal{R}^C$, the soft label matrix of this anchor set can be obtained as $[p(\mathbf{u}_1); \ldots; p(\mathbf{u}_{N_h})] \in \mathbb{R}^{N_h \times C}$. Note that in HAGR, $p(\mathbf{U}) = \mathbf{A}$.

Recall a standard multi-class SSL problem, where a set of labeled datapoints $\mathbf{x}_i$ $(i = 1, \ldots, l)$ with the corresponding discrete labels $y_i \in \{1, \ldots, C\}$ is given and the goal is to estimate the labels of the remaining unlabeled datapoints. Let $\mathbf{Y}_l = [\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_l] \in \mathbb{R}^{l \times C}$ denote the class indicator matrix of the labeled data with $Y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j and $Y_{ij} = 0$ otherwise. Meanwhile, similar to HAGR, we denote $\mathbf{W}$ as the intra-layer adjacency matrix over datapoints, and $\mathbf{Z}^{\mathrm{H}}$ as the cascaded inter-layer adjacency matrix of a hierarchial anchor graph. Then, the regularization framework of learning with an anchor label predictor can be formulated as the following minimization problem:

$$\begin{aligned} \mathrm{argmin}_p \mathcal{Q}_p = &\sum_{i=1}^{l} \|\mathbf{Z}^{\mathrm{H}}_{i\cdot} p(\mathbf{U}) - \mathbf{y}_i\|^2_{\mathrm{F}} + \mu\Omega(p) \\ &+ \frac{\lambda}{2} \sum_{i,j=1}^{N_0} W_{ij} \|\mathbf{Z}^{\mathrm{H}}_{i\cdot} p(\mathbf{U}) - \mathbf{Z}^{\mathrm{H}}_{j\cdot} p(\mathbf{U})\|^2_{\mathrm{F}}. \end{aligned} \tag{8}$$

where $\Omega(\cdot)$ denotes a regularizer on the label predictor, and μ is the corresponding trade-off parameter. As a result, compared with HAGR where a large number of soft labels need to be learned, Eq.8 only needs to optimize a label predictor.

However, the raw representation in the feature space is usually insufficiently powerful to capture the similarity between anchors, and may lead to a poor performance for the following label prediction. Therefore, there are two issues left: (1). how to get effective representations of these coarsest anchors, and (2). how to optimize the corresponding predictor with a faster solution. We address them in Section 4.2 and 4.3, respectively.

4.2 Efficient Estimation of Spectral Representations

In this section, we introduce how to efficiently estimate a discriminative representation of the coarsest anchor set via spectral embedding.

For this purpose, we first consider the issue of constructing an intra-layer adjacency matrix $\tilde{\mathbf{W}}$ over the coarsest anchors, aiming at performing an efficient spectral embedding procedure on these anchors while improving effectiveness. Although there exists a similar method for building the adjacency relationships between these anchors based on an anchor graph [42], our intra-layer adjacency matrix is actually quite different.

First, we determine the relationships of the coarsest anchors with the anchor hierarchy if a hierarchical anchor graph consists of multiple anchor layers. In particular, we employ the cascaded inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$ to estimate the intra-layer weight between the coarsest anchors:

$$\tilde{W}_{ij} = \sum_{s=1}^{N_0} Z^{\mathrm{H}}_{si} Z^{\mathrm{H}}_{sj}. \tag{9}$$

According to Eq.9, once two coarsest anchors share at least one common datapoint based on the cascaded inter-layer adjacency relationships, there will be an intra-layer adjacency edge between them. Meanwhile, if a hierarchical anchor graph only contains an individual anchor layer, this step degrades to the same situation in [42].

The above adjacency relationships can also be expressed in a matrix form:

$$\tilde{\mathbf{W}} = \mathbf{Z}^{\mathrm{H}\top} \mathbf{Z}^{\mathrm{H}} \in \mathbb{R}^{N_h \times N_h}. \tag{10}$$

which can be efficiently computed in $O(N_0 N_h k)$.

Second, we impose a strict sparse constraint on the above intra-layer adjacency matrix. Since Eq.10 accumulates all the nonnegative intra-layer adjacency weights, the obtained adjacency matrix is usually dense, which will slow down the spectral embedding. Therefore, we further prune this intra-layer adjacency matrix by forcing it to be k-sparse:

$$\tilde{W}_{ij} = \begin{cases} \tilde{W}_{ij} & \text{if } \mathbf{u}_j \text{ and } \mathbf{u}_i \text{ are "close"}, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where we define the closeness according to their accumulated adjacency weights rather than the Euclidean distance. Practically, we first retain the adjacency edges with the k largest weights for each coarsest anchor, and then make them undirected. We can thus obtain a sparse and symmetric adjacency matrix, where only the elements corresponding to the anchors with large correlations are reserved.

When two coarsest anchors are close to a classification boundary but locate at different classes, the above pruning operation can remove their intra-layer adjacency relationship and accordingly improve the effectiveness of the intra-layer adjacency matrix. It is understandable that, as the data density between different classes is much lower

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

than that within the same class, the accumulated intra-layer adjacency weight of this suspicious adjacency edge tends to be much smaller than those within the same class. In the experimental section, we will compare the above intra-layer adjacency matrix with the one built upon the RBF kernel under the different settings of graph structures.

Now we look back to the remaining issue of performing the spectral embedding on the coarsest anchors to obtain their representations in a spectral space. Donate $\mathbf{\Sigma}$ as a diagonal degree matrix with $\Sigma_{ii} = \sum_{j=1}^{N_h} \tilde{W}_{ij}$. Then the embedding can be formulated as

$$\operatorname{argmin} \sum_{ij} \tilde{W}_{ij} || \frac{\tilde{\mathbf{u}}_i}{\sqrt{\Sigma_{ii}}} - \frac{\tilde{\mathbf{u}}_j}{\sqrt{\Sigma_{jj}}} ||^2, \tag{12}$$

where $\tilde{\mathbf{u}}_i$ is the spectral representation of the coarsest anchor $\mathbf{u}_i$. According to Eq.12, a pair of anchors with a small intra-layer adjacency weight will have dissimilar spectral representations, and otherwise, their representations tend to be similar.

Let $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1; \ldots; \tilde{\mathbf{u}}_{N_h}] \in \mathbb{R}^{N_h \times d}$ denote the optimized d-dimensional representation of the coarsest anchor set, where each row is the spectral representation of a coarsest anchor. Meanwhile, denote $\mathbf{S} = \mathbf{\Sigma}^{-\frac{1}{2}} \tilde{\mathbf{W}} \mathbf{\Sigma}^{-\frac{1}{2}}$ as the normalized intra-layer adjacency matrix over the coarsest anchors. Then the spectral-representation matrix $\tilde{\mathbf{U}}$ can be obtained according to the following optimization problem:

$$\begin{aligned} &\operatorname{argmin}_{\tilde{\mathbf{U}}} \tilde{\mathbf{U}}^{\mathrm{T}} (\mathbf{I} - \mathbf{S}) \tilde{\mathbf{U}} \\ &\text{s.t.} \tilde{\mathbf{U}}^{\mathrm{T}} \tilde{\mathbf{U}} = \mathbf{I} \end{aligned} \tag{13}$$

which is solved by combing the d smallest eigenvectors of the matrix $(\mathbf{I} - \mathbf{S})$ [31]. Based on Arnoldi technique [21], the time cost of calculating the first d eigenvectors via Eq.13 scales as $O(N_h k d)$, where k is the average number of non-zero entries in each column of this matrix. Of note, the first eigenvectors reflect the main structure of data distributions, and the remaining ones indicate the small difference or the noise. As a result, we can obtain the discriminative spectral representation of this coarsest anchor set with a small number of eigenvectors, which insures its low dimensionality.

When the intra-layer adjacency matrix over the coarsest anchors is "ideal" (which means the coarsest anchors in different classes have zero weights with each other, and those within the same class have large weights with each other), we can employ their spectral representations with C eigenvectors for C-classes, namely, we have $d = C$. Even this adjacency matrix is noisy in most real-world applications and more eigenvectors are required, the dimensionality of the spectral representations can still be much smaller than the size of the anchor set. Later in the experimental section, we will empirically show both the effectiveness and efficiency of the low-dimensional spectral representations .

Note that one can also conduct spectral embedding on the datapoint set or another finer anchor set in a hierarchical anchor graph. Nevertheless, it will lead to a larger computational cost for constructing the corresponding intra-layer adjacency matrix and also slow down the spectral embedding. On the contrary, the spectral representations of the coarsest anchors can be efficiently computed. Besides, as these anchors can roughly cover the data distribution, based on the corresponding intra-layer adjacency matrix, their spectral representations can still be discriminative .

4.3 Faster Optimization with a Linear Predictor

So far we have described how to efficiently estimate the spectral representations of the coarsest anchors. Now we integrate these spectral representations into the regularization framework on a hierarchical anchor graph, and propose the complete FLAG model with a faster optimization.

Let $\tilde{p} : \mathcal{R}^d \to \mathcal{R}^C$ denote the label predictor on the spectral representation of the coarsest anchor set $\tilde{\mathbf{U}} \in \mathbb{R}^{N_h \times d}$. The label matrix of this anchor set can be obtained as $\tilde{p}(\tilde{\mathbf{U}}) \in \mathbb{R}^{N_h \times C}$. To keep computational efficiency, we only consider a linear predictor, represented by $\tilde{\mathbf{P}}$. As such, we can obtain the soft label matrix of the coarsest anchor set:

$$\mathbf{A} = \tilde{\mathbf{U}} \tilde{\mathbf{P}} \in \mathbb{R}^{N_h \times C}. \tag{14}$$

Of note, benefitting from the nonlinear embedding, this simple label predictor can still classify the coarsest anchors with a complex distribution in the feature space.

Then the label matrix on datapoints can be inferred by

$$\mathbf{F} = \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}}, \tag{15}$$

where $\mathbf{Z}^{\mathrm{H}}$ is the cascaded inter-layer adjacency matrix in hierarchical anchor graph models. Compared with pervious anchor-based methods [23], [42], [41], to eventually estimate the labels of unlabeled datapoints, we only need to optimize the above anchor label predictor, of which the size is proportional to the dimensionality of the spectral representation rather than the number of the coarsest anchors.

Now, by integrating the above linear label predictor into the previous regularization framework, Faster Learning on Anchor Graph (FLAG) can be finally formulated as the following optimization problem:

$$\begin{aligned} \operatorname{argmin}_{\tilde{p}} \mathcal{Q}_p = &\sum_{i=1}^{l} \| \mathbf{Z}^{\mathrm{H}}_{i\cdot} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{y}_i \|^2 + \mu || \tilde{\mathbf{P}} ||^2_{\mathrm{F}} \\ &+ \frac{\lambda}{2} \sum_{i,j=1}^{N_0} W_{ij} \| \mathbf{Z}^{\mathrm{H}}_{i\cdot} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Z}^{\mathrm{H}}_{j\cdot} \tilde{\mathbf{U}} \tilde{\mathbf{P}} \|^2. \end{aligned} \tag{16}$$

Denoting $\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}$ as the labeled part of $\mathbf{Z}^{\mathrm{H}}$, we can rewrite Eq.16 into the matrix form:

$$\begin{aligned} \mathcal{Q}_{\tilde{\mathbf{P}}} &= \| \mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Y}_{\mathrm{L}} \|^2_{\mathrm{F}} + \mu \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{P}}) \\ &\quad + \lambda \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} (\mathbf{I} - \mathbf{W}) \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}}) \\ &= \| \mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Y}_{\mathrm{L}} \|^2_{\mathrm{F}} + \mu \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{P}}) + \lambda \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{L}} \tilde{\mathbf{P}}), \end{aligned} \tag{17}$$

where $\tilde{\mathbf{L}}$ is the reduced Laplacian over the spectral representation:

$$\begin{aligned} \tilde{\mathbf{L}} &= \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} (\mathbf{I} - \mathbf{Z}^{0,1} \mathbf{\Lambda}^{-1} \mathbf{Z}^{0,1^{\mathrm{T}}}) \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \in \mathbb{R}^{d \times d} \\ &= \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} - \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}^{0,1} \mathbf{\Lambda}^{-1} \mathbf{Z}^{0,1^{\mathrm{T}}} \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}}. \end{aligned} \tag{18}$$

With simple algebra, the optimization of the label predictor can be computed with a matrix inversion procedure:

$$\tilde{\mathbf{P}} = (\tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}_{\mathrm{L}} \mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} \tilde{\mathbf{U}} + \mu \mathbf{I} + \lambda \tilde{\mathbf{L}})^{-1} \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}_{\mathrm{L}} \mathbf{Y}_{\mathrm{L}}, \tag{19}$$

where the matrix size is equivalent to the dimensionality of the spectral representation d. Compared with HAGR which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 2
Comparison of time complexity of five graph-based methods.

Methods	Graph Construction	Model Optimization
LLGC	$O(N_0 \log N_0 D)$	$O(N_0^3)$
AGR, EAGR	$O(N_0 \log N_1 D)$	$O(N_0 N_1 k + N_1^3)$
HAGR	$O(N_0 \log N_1 D)$	$O(N_0 N_h k + N_h^3)$
FLAG	$O(N_0 \log N_1 D)$	$O(N_0 N_h k + N_h dk + N_h d^2)$

Note that for each method, we have $N_0 > N_1 > N_h \gg d$.

optimizes all the labels of the coarsest anchors via Eq.7, the computation of Eq.19 can be more efficient, as d is practically much smaller than the size of the coarsest anchor set N_h.

Based on the optimized predictor, the soft label matrix on datapoints $\mathbf{F}$ can be estimated by Eqs.14-15. Finally, we can obtain a hard label for any unlabeled datapoint:

$$\widehat{y}_i = \text{argmax}_{r \in \{1,\ldots,c\}} \frac{F_{i,r}}{\beta_r}, i = l+1, \ldots, N_x, \tag{20}$$

where $\beta_r = \mathbf{1}^{\mathrm{T}} \mathbf{F}_{\cdot r}$ is a normalization factor [53], and $\mathbf{F}_{\cdot r}$ is the rth column of $\mathbf{F}$.

4.4 Complexity Analysis

In this section, we summarize the steps of the proposed approach and analyze their time complexities.

(1). Compute inter-layer adjacency matrices to build a hierarchical anchor graph with Eqs.1-2, and construct a sparse intra-layer matrix over the coarsest anchors with Eqs.9-11. The computational cost of the former scales as $O(N_0 \log N_1 D)$, and the latter is $O(N_0 N_h k)$.

(2). Estimate the spectral representations of the coarsest anchors with d eigenvectors, whose cost scales as $O(N_h dk)$.

(3). Calculate the reduced Laplacian in Eq.18 with $O(N_0 kd + N_h d^2)$, and carry out the graph regularization via Eq.19 with $O(d^3)$.

(4). Predict the labels of the coarsest anchors with Eq.14 in $O(N_h dC)$, and infer the labels of datapoints based on Eq.15 in a coarse-to-fine manner, which scales as $O(N_0 kC)$.

Since we usually have $N_h \gg d \gg C$, the total time complexity of FLAG can be simplified to

$$O[N_0(\log N_1 D + N_h k) + N_h d(k+d)].$$

Table 2 lists the time complexities of LLGC (Learning with Local and Global Consistency [50], a typical graph-based SSL method), AGR, EAGR ([42], an improved AGR method), HAGR, and FLAG, in which the ANNS-based kernel regression [41] is applied into all these methods for a fair comparison. From the table we can observe that, although LLGC reduces its graph construction to a linear complexity with respect to the data size, it still faces a cubic cost for its matrix inversion during the optimization. To obtain a high accuracy with more anchors, AGR and EAGR also face a dramatic increase of the computational cost in their model optimization. Then we focus on HAGR and our FLAG. As both of them have the same procedures of computing a series of inter-layer matrices and an intra-layer adjacency matrix (in HAGR means the first term in Eq.6), the comparison of the complexities comes from their rest parts. As we can see from Eq.18 and Eq.6, the remaining costs of calculating the reduced Laplacian in FLAG and HAGR are $O(N_0 kd + N_h d^2)$ and $O(N_0 N_h k)$, respectively. Meanwhile, rather than conducting the matrix inversion with a cost of $O(N_h^3)$, FLAG implements a spectral embedding procedure with a linear complexity of $O(N_h dk)$ on the pruned adjacency matrix, and its optimization only involves the matrix inversion on a $d \times d$ matrix with a cubic cost of $O(d^3)$. Since d and k are much smaller than N_h, we have $O(N_0 kd + N_h kd + N_h d^2 + d^3) \ll O(N_0 N_h k + N_h^3)$. As a result, FLAG has a much less cost than HAGR and other conventional fast learning approaches, and can efficiently deal with large-scale datasets with more coarsest anchors.

5 Experiment

Now we investigate both the effectiveness and efficiency of FLAG on real-world datasets. All the experiments are implemented on a PC with E5-2620 v2 @2.10 GHz and 64G RAM. Here we use the following five datasets with sizes varying from thousands to millions. The descriptions of these datasets are given in below, and some statistics of them are listed in Table 3.

(1) *FaceMIT:* The FaceMIT data set [3] contains 6,977 training samples (2,429 positives, 4,548 negatives) and 24,045 testing samples (472 positives, 23,573 negatives). In our experiment, we only employ the training set for binary classification.

(2) *Newsgroup:* A tiny version of the 20newsgroups data, with binary occurance data for 100 words across 16,242 postings. It is tagged the postings by the highest level domain in the array 'newsgroups' [34].

(3) *Extended MNIST:* This dataset is widely used in many large-scale graph-based works [14], [19], [24]. The original MNIST dataset contains 70,000 samples of handwritten digits from '0' to '9' [20]. Each of the ten classes contains about 7,000 samples, which are images centered in a 28×28 field by computing the center of mass of the pixels. This extended dataset is constructed by translating the original images in MNIST one pixel in each direction. As a result, there are 630,000 samples in 900 dimensions by using the normalized grayscale values as features.

(4) *Extended USPS:* The USPS dataset is for hand-written digits recognition and contains 7,291 training samples and 2,007 test samples of digit images. All the digits in the training set of USPS are extended by shifting the 16×16 images in all directions for up to five pixels [38]. There are 882,211 samples in 676 dimensions in total. We directly use the normalized grayscale values as features.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 3
Details of the five databases used in our experiments.

	FaceMIT	Newsgroup	Extended MNIST	Extended USPS	MNIST8M
# of instances	6,977	16,242	630,000	882,211	8,100,000
# of categories	2	4	10	10	10
# of dimensions	361	100	900	676	784

(5) *MNIST8M:* This dataset has been used in [15], [22], [25] to verify the efficiency of large-scale learning algorithms. It contains totally 8,100,000 samples in 784 dimensions.

For convenience, these datasets are categorized into small, medium and large sizes. Specifically, we regard FaceMIT, and Newsgroup as small-size datasets, Extended MNIST, and Extended USPS as medium-size datasets, and MNIST8M as a large-size dataset.

5.1 Comparison to State-of-the-art Approaches

We compare FLAG with several state-of-the-art fast learning approaches, such as AGR, EAGR and HAGR, to demonstrate its efficiency and effectiveness. We also report the performances of two baseline methods including 1NN, and linear SVM. The methods for comparison are described in below.

(1) 1NN: It determines the label of an instance by referring to its nearest datapoint in the labeled set.

(2) LSVM [11]: We use the Linear SVM implementation from LIBLINEAR, which is a library for large-scale linear classification.

(3) LLGC [50]: This typical graph-based method directly optimizes the labels of datapoints. In our experiments, we employ a kNN strategy for its graph construction.

(4) AGR-N_1 [23]: It is the original anchor-graph-based learning method with an individual anchor layer ($h = 1$), where N_1 is the number of anchors. Of note, this method can be viewed as a reduced version of HAGR.

(5) EAGR-N_1 [42]: Compared with AGR-N_1, it remodifies the regularizer by constructing a smoothness constraint on the labels of anchors.

(6) FLAG-N_1: Compared with AGR-N_1, it first imposes a strict sparse constraint on the intra-layer adjacency matrix over anchors to obtain their spectral representations, and then introduces a linear predictor to estimate the labels of these anchors. As a simplest version of the FLAG method, it is proposed to verify the efficiency of learning an anchor label predictor for small-size datasets and show the effectiveness of adding a finer anchor layer for larger-size datasets.

(7) HAGR-N_1-...-N_h [41]: This HAGR approach is built upon a hierarchical anchor graph with $h(h > 1)$ anchor layers, where $N_b(b = 1, \ldots, h)$ denotes the scale of the b-th anchor layer. Compared with method (4), it infers the labels of datapoints in a coarse-to-fine manner.

(8) FLAG-N_1-...-N_h: Different from method (7), it integrates an anchor label predictor with the hierarchial label inference to estimate the labels of datapoints. Meanwhile, compared with method (6), the spectral representations of the coarsest anchors here are obtained based on their sparse intra-layer adjacency matrix, which is built upon the anchor hierarchy. This approach is designed for classification on medium-size and large-size datasets.

For a fair comparison, the kernel widths (δ) in above approaches are set by cross validation, and the trade-off parameters (λ and μ) are tuned to their optimal values. Besides, we empirically choose sparse parameters (k) from 2 to 6 for all approaches.

5.1.1 Small-Size Datasets

We first conduct experiments on FaceMIT and Newsgroup. As the sizes of these datasets are small, we only build anchor graphs with an individual anchor set, where N_1=2,000 and N_1=3,000 are used for FaceMIT and Newsgroup, respectively. We perform FLAG on these anchor graphs with the optimized d chosen from 1 to 30. We vary the number of labeled samples in $\{2, 4, \ldots, 10\}$ per class. The average classification accuracies over 20 trials are reported in Table 4 and 5, where the time costs with 10 labels per class are listed at the last column.

From the tables, we obtain the following observations. **First**, the accuracies of all graph-based SSL approaches stay at a higher level than those of LSVM and 1NN, especially when the number of labeled datapoints is small, which demonstrates the importance of leveraging unlabeled data in SSL. **Second**, benefitting from anchor-based label inference, both AGR-N_1 and EAGR-N_1 reduce the time cost to a much lower level than LLGC. However, compared with LLGC, their classification accuracies can be worse in some cases. **Third**, FLAG-N_1 performs SSL with the smallest time cost, and consistently obtains higher accuracies than other three approaches. These results demonstrate the superiority of FLAG-N_1 for scaling up graph-based learning.

5.1.2 Medium-Size Datasets

For two medium-size datasets, we first follow [23] and perform PCA to reduce the original image dimensions to 86. Then for each of them, we construct two anchor graphs with N_1 =5,000 and N_1=20,000, and one hierarchical anchor graph with N_1=200,000, N_2=20,000. We perform FLAG on these graphs with the optimized d chosen from 10 to 300. The number of labeled samples varies from $\{2, 4, \ldots, 10\}$ and $\{20, 40, \ldots, 100\}$ per class for Extended MNIST and Extended USPS, respectively. The average classification accuracies over 20 trials are shown in Table 6 and 7, and the time costs are reported at the last column.

From the results in these tables, the following observations can be made. **First**, compared with other approaches under the same anchor configurations, FLAG consistently obtains better performances with less time costs. This result demonstrates both the efficiency and effectiveness of our label predictor optimization. **Second**, anchor-graph-based approaches with N_1 =20,000 almost obtain higher

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 4
The comparison of different approaches on FaceMIT.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	55.36±10.66	64.91±10.26	68.29±7.45	68.92±5.59	72.36±5.45	0.03
LSVM	50.21±16.32	56.73±15.82	64.44±14.44	69.35±10.18	73.72±9.40	0.11
LLGC	71.43±12.23	83.37±8.85	85.36±4.59	88.11±4.56	89.62±3.73	8.96
AGR-2,000	68.06±14.18	77.65±10,81	82.18±5.86	86.86±5.18	87.60±5.88	0.85
EAGR-2,000	66.11±14.23	79.18±9.58	83.37±6.50	85.31±5.97	88.66±4.31	0.76
FLAG-2,000	**80.43±14.41**	**86.06±10.19**	**90.28±3.78**	**91.33±1.68**	**91.96±1.90**	0.54

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 5
The comparison of different approaches on Newsgroup.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	33.30±6.12	37.26±5.69	39.46±5.65	40.57±6.03	40.91±5.61	0.04
LSVM	40.42±5.77	48.07±3.86	53.32±4.29	57.26±4.44	60.43±3.09	0.01
LLGC	52.12±6.67	57.48±3.75	59.85±2.92	60.02±3.23	61.41±2.42	85.73
AGR-3,000	54.55±3.70	59.93±5.48	61.79±3.72	62.82±3.21	63.91±2.85	1.98
EAGR-3,000	54.36±6.86	59.39±6.55	61.67±4.39	62.69±3.00	63.21±2.69	1.85
FLAG-3,000	**60.28±7.06**	**66.43±3.01**	**68.80±2.01**	**69.01±1.62**	**69.67±1.67**	1.33

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 6
The comparison of different approaches on Extended MNIST.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	42.21±3.45	49.11±3.26	54.55±2.85	57.56±2.16	60.04±1.70	2.56
LSVM	45.74±2.49	51.96±3.13	56.44±2.65	59.78±2.77	61.79±1.96	3.15
AGR-5,000	79.47±2.20	83.11±1.89	85.77±2.20	87.53±0.95	88.53±0.82	7.98
EAGR-5,000	79.76±1.86	83.82±1.91	86.92±1.37	88.72±0.60	89.48±0.73	7.24
FLAG-5,000	85.15±2.70	89.27±3.00	91.24±1.59	92.40±0.39	92.57±0.29	5.81
AGR-20,000	79.15±2.49	84.05±2.60	86.90±1.82	89.27±1.35	90.17±1.31	151.62
EAGR-20,000	79.18±2.51	84.41±2.26	87.52±1.64	89.98±1.10	90.86±1.10	150.04
FLAG-20,000	81.66±3.21	89.48±2.61	91.36±1.67	93.17±0.93	94.05±0.24	7.91
HAGR-200,000-20,000	83.01±2.18	87.46±1.93	90.04±1.42	91.66±0.92	92.62±0.46	156.83
FLAG-200,000-20,000	**87.78±1.97**	**92.42±1.83**	**94.01±1.38**	**95.02±0.39**	**95.07±0.34**	14.98

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 7
The comparison of different approaches on Extended USPS.

Num of labels per class	20	40	60	80	100	Time cost (in second)
1NN	45.94±1.38	56.58±0.52	62.03±0.58	66.40±0.56	69.25±0.51	3.65
LSVM	22.57±1.40	24.73±1.01	25.55±0.52	26.94±0.59	27.52±1.19	4.75
AGR-5,000	73.13±1.37	78.98±0.82	81.48±0.83	83.61±0.45	84.63±0.32	9.49
EAGR-5,000	76.19±1.38	82.29±0.59	83.63±0.69	85.90±0.48	86.58±0.40	8.65
FLAG-5,000	80.40±1.83	84.84±0.23	86.12±0.63	87.01±0.48	87.70±0.24	7.35
AGR-20,000	75.69±1.01	81.57±1.42	83.92±1.61	86.35±0.75	87.20±0.48	159.90
EAGR-20,000	77.61±1.27	84.09±0.95	86.11±0.67	88.18±0.44	88.89±0.27	158.48
FLAG-20,000	80.55±1.55	86.39±0.84	88.22±0.80	89.87±0.65	90.71±0.44	21.87
HAGR-200,000-20,000	79.92±1.08	85.87±0.96	87.85±0.59	89.78±0.66	90.65±0.71	167.68
FLAG-200,000-20,000	**84.71±1.50**	**89.52±0.41**	**90.68±0.40**	**91.53±0.33**	**92.07±0.48**	28.91

The best results are shown in **bold**. The last column shows time costs with 100 labeled data per class.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 8
The comparison of different approaches on MNIST8M.

Num of labeled per class	2	4	6	8	10	Time cost (in second)
1NN	41.80±3.44	51.89±3.58	56.15±2.41	59.57±2.14	61.87±1.28	35.13
LSVM	45.08±4.19	54.69±2.45	56.23±2.52	59.50±1.45	61.66±1.19	39.82
AGR-5,000	76.53±4.15	81.99±2.98	83.22±1.66	83.94±1.94	85.02±1.48	81.26
EAGR-5,000	79.46±4.02	84.84±4.02	86.63±2.39	87.24±2.87	88.48±1.59	80.09
FLAG-5,000	81.25±4.40	86.92±1.36	89.14±1.63	89.77±1.25	90.72±0.90	76.25
AGR-30,000	79.50±3.57	86.83±3.14	89.26±2.39	89.92±1.96	90.54±0.97	663.75
EAGR-30,000	79.77±3.19	87.17±2.45	89.64±1.93	90.27±1.25	90.91±0.89	661.31
FLAG-30,000	81.55±4.11	87.51±2.63	89.84±1.44	90.46±1.47	91.85±0.89	102.41
HAGR-300,000-30,000	83.24±3.89	88.29±1.86	89.87±1.79	90.73±1.45	91.96±1.09	705.68
HAGR-300,000-30,000-5,000	82.27±3.74	87.50±2.07	89.39±2.27	90.32±1.78	91.49±1.16	137.15
FLAG-300,000-30,000	**85.84±4.21**	**91.12±1.92**	**92.24±1.19**	**92.77±0.69**	**93.40±0.20**	134.58

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

accuracies than those with N_1=5,000, which shows the importance of employing a large anchor set. **Third**, by adding a larger anchor layer, HAGR-200,000-20,000 achieves higher classification accuracies than AGR-20,000. However, its performances are only comparable or even worse than those of FLAG-5,000 on Extended MNIST and FLAG-20,000 on Extended USPS. **Forth**, based on the anchor hierarchy, FLAG-200,000-20,000 improves its adjacency relationships and consistently outperforms all other methods with the efficient implementation.

5.1.3 Large-Size Dataset

We further test the scalability of FLAG on a large-size dataset, namely MNIST8M. In particular, we build two anchor graphs with N_1=5,000 and N_1=30,000. Besides, we construct two hierarchical anchor graphs with N_1=300,000, N_2=30,000 and N_1=300,000, N_2=30,000, N_3=5,000. By repeating the similar evaluation process, we report the average classification accuracies over 10 trials in Table 8.

Similar to the results of the above experiments, we can see that, based on the same anchor configuration, FLAG consistently outperforms other fast learning approaches with different numbers of labeled samples in terms of both the efficiency and effectiveness. We also observe that, by adding a small anchor layer, the time cost of HAGR-300,000-30,000-5,000 is much smaller than that of HAGR-300,000-30,000. However, the former sacrifices the accuracy at the same time, and even results in slightly worse performances than FLAG-30,000. In contrast, FLAG-300,000-30,000 obtains much higher accuracies with less time costs.

It is worthwhile to note that in all the experiments above, the accuracy of FLAG is much higher than that of HAGR with the same graph structure, especially when the number of the labeled data is small. The main reason is that, the HAGR classifier employs the coarsest-anchor-based coding as its input feature and requires the size of this anchor set N_h to be large for well capturing the data distribution. In contrast, our FLAG classifier employs the spectral representation with the dimensionality d as its input feature, which can keep its hypothesis space in a lower level while enlarging the scale of the coarsest anchor set. As both of them actually assign labels based on linear decision surfaces with the corresponding feature, their VC dimensions can be easily obtained [28], namely $N_h + 1$ for HAGR and $d + 1$ for FLAG. According to the computational learning theory [5], to approximately learn a target function in a hypothesis space, the number of required labeled datapoints is linear with its VC dimension. Therefore, compared with HAGR built upon the same hierarchical anchor graph, FLAG requires less labeled data to well train the model itself. As our work aims at the setting of semi-supervised learning where only a few data are labeled, FLAG is a more powerful approach to handle the large scale classification in terms of both the effectiveness and efficiency.

5.2 On the Improvements of FLAG

So far, we can see that FLAG improves both the efficiency and effectiveness of anchor-graph-based learning by optimizing an anchor label predictor on the spectral representation. In this section, we further investigate how these improvements are obtained based on the following aspects: (1). the construction of the intra-layer adjacency matrix over the coarsest anchors, and (2). the regularizer on the label predictor.

For simplicity, we follow the settings in [37] and use the dataset MNIST [20] for binary classification under two settings: (1). the first 5 versus the last 5 digits in MNIST1, and (2). the odd digits versus the even digits in MNIST2.

5.2.1 On the Intra-layer Adjacency Matrix Construction

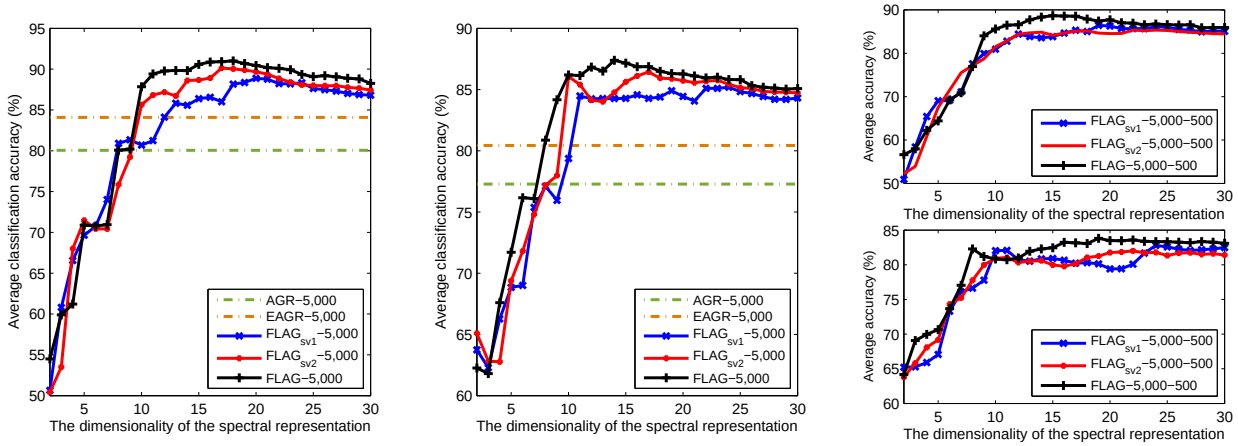
In order to verify the first aspect, we define two intermediate methods for a better comparison:

FLAG_{sv1}: Compared with FLAG, this simplified version does not implement the pruning operation on the accumulated intra-layer matrix over the coarsest anchors. In other words, it directly obtains their spectral representations by performing spectral embedding with the original accumulated intra-layer matrix.

FLAG_{sv2}: Compared with FLAG, this simplified version first constructs an intra-layer adjacency matrix over the coarsest anchors with the RBF kernel and keeps the top k values for each anchor. Then, it estimates the spectral representations of these anchors by performing spectral embedding with the adjacency matrix.

We fix μ=0 and optimize λ for a fair comparison. For all the compared anchor-graph-based methods, we first

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(a) MNIST1 (b) MNIST2 (c) MNIST1(top) and MNIST2(bottom)

Fig. 3. The accuracy variation with respect to the dimensionality of the spectral representation. Note the approaches in (a) and (b) are built upon a hierarchical anchor graph ($h = 1$) with 5,000 anchors, and those in (c) are built upon a hierarchical anchor graph ($h = 2$) with 5,000 and 500 anchors.

TABLE 9
The comparison of time costs (in seconds) of different approaches (without the time cost of graph construction). Note that the cost of calculating an intra-layer adjacency matrix over anchors, i.e., $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$, is nearly 0.05s.

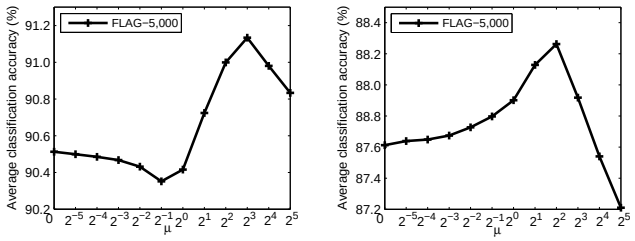
Procedure	step.1	step.2	step.3	step.4	
N_1=5,000	spectral representation estimation	reduced Laplacian calculation	matrix inversion	label inference	total time cost
AGR-5,000	-	0.30	3.06	0.01	3.37
EAGR-5,000	-	0.08	3.06	0.01	3.15
$\mathrm{FLAG}_{\mathrm{sv1}}$-5,000 ($d$=18)	1.37	0.07	0.01	0.01	1.46
$\mathrm{FLAG}_{\mathrm{sv2}}$-5,000 ($d$=18)	0.20	0.07	0.01	0.01	0.29
FLAG-5,000 (d=18)	0.18	0.07	0.01	0.01	0.27

employ the same anchor graph model with 5,000 anchors. We randomly sample 20 datapoints as the labeled part and keep the rest unlabeled for the setting of SSL. The average accuracy curves of AGR-5,000, EAGR-5,000, and $\mathrm{FLAG}_{\mathrm{sv1}}$-5,000, $\mathrm{FLAG}_{\mathrm{sv2}}$-5,000, FLAG-5,000 with the varying dimensionality of the spectral representation are shown in Fig.3, and the time costs of different approaches (without the time cost of graph construction) are listed in Table 9.

From the Fig.3, we can obtain the following observations. **First**, by introducing a linear label predictor on the spectral representations of anchors, FLAG-5,000 and $\mathrm{FLAG}_{\mathrm{sv1}}$-5,000, $\mathrm{FLAG}_{\mathrm{sv2}}$-5,000 can obtain better performances than AGR-5,000. **Second**, benefitting from the strict sparse constraint on the intra-layer adjacency matrix over anchors, FLAG-5,000 obtains higher classification accuracies than $\mathrm{FLAG}_{\mathrm{sv1}}$-5,000. This result shows that the pruning operation can remove most of suspicious edges and improve the effectiveness of the intra-layer adjacency relationships, which is also consistent with the observation that sparse graphs perform better than dense graphs [52]. **Third**, although $\mathrm{FLAG}_{\mathrm{sv2}}$-5,000 employs a sparse intra-layer matrix over the coarsest anchors for estimating their spectral representations as well, its best accuracy is still worse than that of FLAG-5,000. The main reason is that, when two anchors are close to a classification boundary but locate at different classes, the pruning operation in FLAG can remove this kind of intra-layer adjacency edges as we mentioned in Section 4.2, which accordingly improves effectiveness. In contrast, the intra-layer adjacency relationships built upon the RBF kernel in $\mathrm{FLAG}_{\mathrm{sv2}}$ completely depend on the Euclidean distance, and the above noisy adjacency edges are hard to be discovered and filtered from the intra-layer adjacency matrix.

Taking into account the time cost in Table 9, the following observations can be obtained. **First**, the time cost of EAGR-5,000 is slightly smaller than that of AGR-5,000 (HAGR with $h = 1$). It is understandable that when calculating the reduced Laplacian matrix, AGR involves multiple times sparse matrix multiplication and EAGR only involves the operation one time [42]. **Second**, FLAG-5,000 and $\mathrm{FLAG}_{\mathrm{sv2}}$-5,000 are faster than $\mathrm{FLAG}_{\mathrm{sv1}}$-5,000, as the time complexity of the spectral embedding with a sparse matrix is much smaller than the one with a dense matrix. **Third**, all three FLAG-based versions carry out the inversion of a smaller-size matrix, and are therefore more efficient than the traditional fast learning approaches, especially AGR (HAGR). It is worthwhile to note that when the number of the coarsest anchors increases for better capturing the data distribution, the improvement of the efficiency will be more significant.

Moreover, for three FLAG-based versions, namely FLAG, $\mathrm{FLAG}_{\mathrm{sv1}}$, and $\mathrm{FLAG}_{\mathrm{sv2}}$, we additionally compare their performances based on a hierarchical anchor graph, in order to further demonstrate the effectiveness of our intra-



(a) MNIST1 (b) MNIST2

Fig. 4. The accuracy variation with respect to the parameter μ.

layer adjacency matrix construction. For this purpose, we repeatedly construct two-anchor-layer graphs for 10 times with the sizes of 5,000 and 500, respectively. By 10 times randomly sampling for building labeled data, the average accuracies of three approaches built upon these graphs are shown in Fig.3(c). As we can see, the best accuracies of FLAG-5,000-500 are still higher than those of FLAG_{sv1}-5,000-500 and FLAG_{sv2}-5,000-500 under two settings.

5.2.2 On the Predictor Regularizer

Finally we test the sensitivity of the weighting parameter μ on the predictor regularizer. We set other parameters to the optimized values according to the experiments above. Then, we vary μ and the average classification accuracies over 10 trials are shown in Fig.4. As we can see, compared with the accuracy where μ=0, this proposed regularizer can further improve the performance of the anchor label predictor and the classification accuracy stays at a high level over a wide range of the parameter variation.

6 Conclusion and Future Work

This work introduces a novel approach called Faster Learning on Anchor Graph (FLAG), which further scales up anchor-graph-based models and meanwhile improves their effectiveness. In FLAG, the labels of the coarsest anchors are obtained by learning a linear predictor on their low-dimensional spectral representations, which can be efficiently estimated based on a proposed sparse intra-layer adjacency matrix over these anchors. To optimize the anchor label predictor, we also develop a novel regularization framework based on a hierarchical anchor graph. In this way, the optimization can be computed with a faster matrix inversion procedure, where the matrix size is only equivalent to the dimensionality of the spectral representation. Furthermore, owing to the flexible structure of hierarchical anchor graph models, FLAG can be scaled to different scales of datasets, including large-scale ones. The experiments on publicly available datasets of various sizes have demonstrated this superiority over the conventional fast learning models.

Finally, we discuss the possible research direction in the future. In this work, we only employ a linear label predictor on the spectral representation to keep computational efficiency. Differently, we may use more complex label prediction models, such as deep neural networks. In this way, we can build a hierarchical anchor graph model upon a deep neural network, which leads to a semi-supervised deep neural network training approach for large-scale data.

Acknowledgments

The authors sincerely appreciate the useful comments and suggestions from the anonymous reviewers. This work was partially supported by the National 973 Program of China under grant 2014CB347600, and the National Nature Science Foundation of China under grant 61432019, and 61772171.

References

[1] R. K. Ando and T. Zhang, "Learning on graph with laplacian regularization," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 25–32.

[2] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[3] M. C. F. Biological and C. Learning, "Cbcl face database #1," http://www.ai.mit.edu/projects/cbcl.

[4] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the conference on Computational Learning Theory*. ACM, 1998, pp. 92–100.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *Journal of the ACM*, vol. 36, no. 4, pp. 929–965, 1989.

[6] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recognition Letters*, vol. 16, no. 1, pp. 105–111, 1995.

[7] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 203–233, 2008.

[8] J. Chen, H.-r. Fang, and Y. Saad, "Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection," *Journal of Machine Learning Research*, vol. 10, no. Sep, pp. 1989–2012, 2009.

[9] L. Chen, I. W. Tsang, and D. Xu, "Laplacian embedded regression for scalable manifold regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 902–915, 2012.

[10] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the International Conference on World Wide Web (WWW)*. ACM, 2011, pp. 577–586.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, no. 9, pp. 1871–1874, 2008.

[12] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, 2001, vol. 1.

[13] A. Goyal, H. Daumé III, and R. Guerra, "Fast large-scale approximate graph construction for nlp," in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 1069–1080.

[14] M. Hein and S. Setzer, "Beyond spectral clustering-tight relaxations of balanced graph cuts," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 2366–2374.

[15] C.-J. Hsieh, S. Si, and I. S. Dhillon, "A divide-and-conquer solver for kernel support vector machines." in *Proceedings of the International Conference on Machine learning (ICML)*, 2014, pp. 566–574.

[16] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix." in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015, pp. 687–696.

[17] L. Jiang, P. Luo, J. Wang, Y. Xiong, B. Lin, M. Wang, and N. An, "Grias: an entity-relation graph based framework for discovering entity aliases," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 2013, pp. 310–319.

[18] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the International Conference on Machine learning (ICML)*, vol. 99, 1999, pp. 200–209.

[19] M. Karlen, J. Weston, A. Erkan, and R. Collobert, "Large scale manifold transduction," in *Proceedings of the International Conference on Machine learning (ICML)*. ACM, 2008, pp. 448–455.

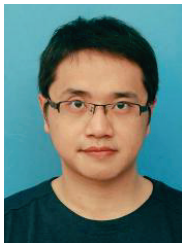
[20] Y. Lecun, "Mnist," http://yann.lecun.com/exdb/mnist/.

[21] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Society for Industrial and Applied Mathematics (SIAM), 1998.

[22] M. Li, J. T.-Y. Kwok, and B. Lü, "Making large-scale nyström approximation possible," in *Proceedings of the International Conference on Machine learning (ICML)*, 2010, pp. 631–638.
[23] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 679–686.
[24] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2624–2638, 2012.
[25] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," *Large Scale Kernel Machines*, pp. 301–320, 2007.
[26] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014.
[27] S. Melacci and M. Belkin, "Laplacian Support Vector Machines Trained in the Primal," *Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, 2011.
[28] T. M. Mitchell, *Machine learning*. McGraw-Hill Science, 1997.
[29] T. Mu, J. Y. Goulermas, J. Tsujii, and S. Ananiadou, "Proximity-based frameworks for generating embeddings from multi-output data," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2216–2232, 2012.
[30] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.
[31] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, vol. 14, no. 2, 2001, pp. 849–856.
[32] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *Proceedings of the International Semantic Web Conference (ISWC)*, 2013, pp. 542–557.
[33] D. Rao and D. Yarowsky, "Ranking and semi-supervised classification on large scale graphs using map-reduce," in *Proceedings of the Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics, 2009, pp. 58–65.
[34] S. Roweis, "Newsgroup," http://www.cs.nyu.edu/~roweis/data.html.
[35] W. Shen, J. Wang, P. Luo, and M. Wang, "A graph-based approach for ontology population with named entities," in *Proceedings of the ACM International Conference on Information and knowledge management*. ACM, 2012, pp. 345–354.
[36] A. Singhal, "Introducing the knowledge graph: things, not strings," *Official google blog*, 2012.
[37] I. W. Tsang and J. T. Kwok, "Large-scale sparsified manifold regularization," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1401–1408.
[38] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. 1, pp. 363–392, 2005.
[39] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-nn graph construction for visual descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 1106–1113.
[40] J. Wang and Y. Xia, "Fast graph construction using auction algorithm," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012, pp. 873–882.
[41] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1101–1114, 2017.
[42] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1864–1877, 2016.
[43] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 733–746, 2009.
[44] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Transactions on Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.
[45] Z.-J. Zha, T. Mei, J. Wang, Z. Wang, and X.-S. Hua, "Graph-based semi-supervised learning with multiple labels," *Journal of Visual Communication and Image Representation*, vol. 20, no. 2, pp. 97–103, 2009.
[46] 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 (ICML)*, 2009, pp. 1233–1240.
[47] K. Zhang, L. Liang, J. T. Kwok, and S. Vucetic, "Scaling up graph-based semi-supervised learning via prototype vector machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 444–457, 2015.
[48] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: A semi-supervised co-training algorithm," in *Proceedings of the international ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 73–82.
[49] Y.-M. Zhang, K. Huang, G. Geng, and C.-L. Liu, "Fast knn graph construction with locality sensitive hashing," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 660–674.
[50] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 321–328.
[51] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1601–1608.
[52] X. Zhu, "Semi-supervised learning literature survey," *Technical report, University of Wisconsin Madison*, 2005.
[53] 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 (ICML)*, 2003, pp. 912–919.
[54] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

Weijie Fu is currently working toward the PhD degree in the School of Computer and Information, Hefei University of Technology(HFUT). His current research interest focuses on machine learning and data mining.

Meng Wang is a professor at the Hefei University of Technology, China. He received his B.E. degree and Ph.D. degree in the Special Class for the Gifted Young and the Department of Electronic Engineering and Information Science from the University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, journal and conference papers in these areas. He is the recipient of the ACM SIGMM Rising Star Award 2014. He is an associate editor of IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), IEEE Transactions on Circuits and Systems for Video Technology (IEEE TCSVT), and IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS).



Shijie Hao is an associate professor at the Hefei University of Technology(HFUT), China. He received his B.E., M.S. and Ph.D. Degree in the School of Computer and Information from HFUT. His current research interests include machine learning and image processing.

Tingting Mu received the B.Eng. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the Ph.D. degree in electrical engineering and electronics from the University of Liverpool, Liverpool, U.K., in 2008. She is currently a Lecturer with the School of Computer Science, University of Manchester, Manchester, U.K. Her current research interests include machine learning, data visualization, and mathematical modeling, with applications to information retrieval, text mining, and bioinformatics.