Learning on Big Graph: Label Inference and Regularization with Anchor Hierarchy

Meng Wang, *Member, IEEE*, Weijie Fu, Shijie Hao, Hengchang Liu, and Xindong Wu, *Fellow, IEEE*

Abstract—Several models have been proposed to cope with the rapidly increasing size of data, such as Anchor Graph Regularization (AGR). The AGR approach significantly accelerates graph-based learning by exploring a set of anchors. However, when a dataset becomes much larger, AGR still faces a big graph which brings dramatically increasing computational costs. To overcome this issue, we propose a novel Hierarchical Anchor Graph Regularization (HAGR) approach by exploring multiple-layer anchors with a pyramid-style structure. In HAGR, the labels of datapoints are inferred from the coarsest anchors layer by layer in a coarse-to-fine manner. The label smoothness regularization is performed on all datapoints, and we demonstrate that the optimization process only involves a small-size reduced Laplacian matrix. We also introduce a fast approach to construct our hierarchical anchor graph based on an approximate nearest neighbor search technique. Experiments on million-scale datasets demonstrate the effectiveness and efficiency of the proposed HAGR approach over existing methods. Results show that the HAGR approach is even able to achieve a good performance within 3 minutes in an 8-million-example classification task.

Index Terms—Semi-supervised learning, graph-based learning, label smoothness regularization, label inference

1 Introduction

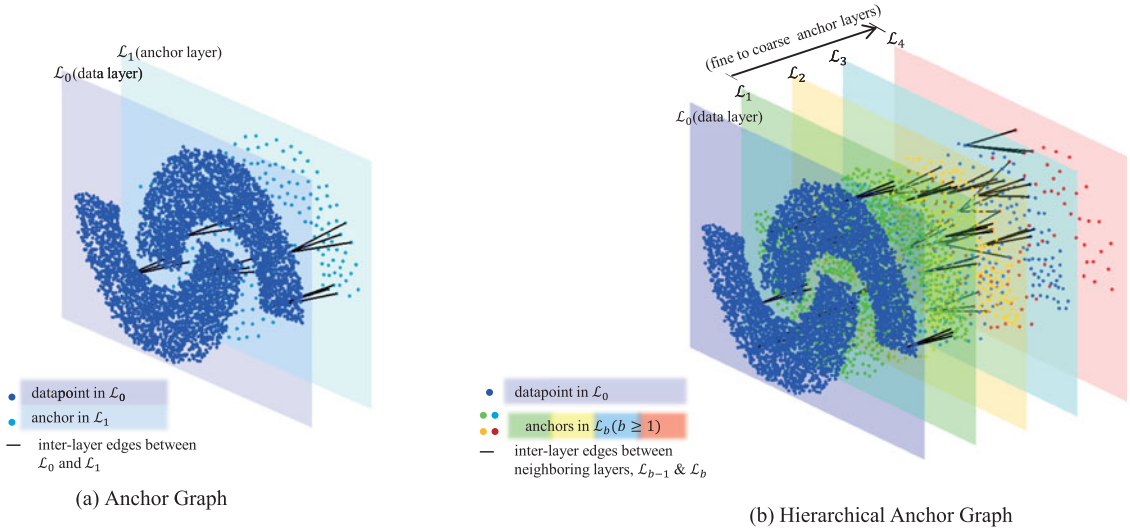
Semi-supervised learning (SSL) methods [51], which exploit the prior knowledge from unlabeled data to improve classification performance, have been widely used to handle datasets where only a portion of data are labeled. Most of these methods are developed based on the cluster assumption [47] or the manifold assumption [1]. The former supposes that nearby points are likely to have the same label, while the latter assumes that each class lies on a separate low-dimensional manifold embedded in a higher dimensional space. In recent years, various semi-supervised learning methods have been developed under these assumptions, including mixture methods [4], co-training [2], semi-supervised support vector machines [18], and graph-based methods [50].

In this paper, we focus on the family of graph-based semi-supervised learning (GSSL) methods, where the label dependencies among datapoints are captured by a weighted graph. These methods first construct adjacency relationships between all datapoints and then propagate labels from labeled data to unlabeled data with the above adjacency edges. Since many forms of real-world data, such as handwritten digits, faces, medical data, and speech data, exhibit such a kind of intrinsic graph structure, GSSL has been applied to many applications, and achieves satisfying performance [10], [41], [42]. Meanwhile, this roadmap can be extended to building other advanced graph models, such as hypergraph [17], [48] and multi-graph [7], [37], to describe more complex relationships among real-world entities like multimodal media contents [12], [13], [27].

In spite of the progress made in recent years, most GSSL methods remain challenging mainly due to their cubic complexity in optimization. Facing the ever increasing data size, these approaches tend to be inefficient in dealing with large-scale datasets. To address this issue, recent works seek to employ anchors in scaling up graph-based learning models, such as Anchor Graph Regularization (AGR) [23], and Efficient Anchor Graph Regularization (EAGR) [36] (for simplicity, we call both of them AGR without differentiation except in comparative experiments). In these models, anchors refer to the points that roughly cover the data distribution. AGR then builds an anchor graph to model the inter-layer adjacency between the data layer and the anchor layer. For clarity, a few inter-layer edges in the anchor graph built on a two-moon dataset are shown in Fig. 1a. The efficiency of these approaches lies in two steps: 1) they build the intra-layer adjacency relationships based on the anchors, instead of computing all pair-wise adjacencies between the datapoints in an exhaustive way; and 2) they infer the labels of datapoints from the anchors based on their inter-layer adjacency relationships. As the number of anchors can be much smaller than datapoints, both the graph construction and the learning process become much faster than those in traditional graph-based approaches. However, to obtain a reasonable accuracy, anchors need to be sufficiently dense in order to build effective adjacency relationships.

- *M. Wang, W. Fu, and S. Hao are with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China. E-mail: {eric.mengwang, fwj.edu, hfut.hsj}@gmail.com.*
- *H. Liu is with the Department of Computer Science, University of Science and Technology of China, Hefei, Shi 230022, China. E-mail: hcliu@ustc.edu.cn.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, and the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70504. E-mail: xwu@louisiana.edu.*

Manuscript received 4 Sept. 2016; revised 8 Nov. 2016; accepted 7 Jan. 2017. Date of publication 17 Jan. 2017; date of current version 30 Mar. 2017. Recommended for acceptance by H.T. Shen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2654445



(a) Anchor Graph (b) Hierarchical Anchor Graph

Fig. 1. An illustrative example of anchor graph (consisting of 5,000 datapoints and an anchor layer with 250 anchors) and hierarchical anchor graph (consisting of 5,000 datapoints, and multiple anchor layers with 1,000, 500, 250, and 100 anchors, respectively). For simplify, only a tiny fraction of inter-layer edges are shown.

Therefore, when dealing with extremely large-scale datasets, the computational costs of existing anchor-graph-based approaches will dramatically increase and become even practically intractable. One possible way is to use only a small number of anchors, but too sparse anchors will degrade performance as the label inference and the label smoothness regularization cannot be performed reliably.

To address this issue, in this paper, we introduce a novel hierarchical anchor graph and develop a scalable SSL approach named Hierarchical Anchor Graph Regularization (HAGR). Different from the existing graphs, our proposed graph contains multiple layers of anchors in a pyramid-style structure. It consists of a layer of original datapoints and multiple layers of anchors that describe the original datapoints from fine to coarse, as illustrated in Fig. 1b. Based on the proposed graph model, we infer the labels of datapoints from the coarsest anchors layer by layer based on inter-layer adjacency relationships. Although the label smoothness regularization is performed on all datapoints, we demonstrate that the optimization only involves a reduced Laplacian matrix with the size of the coarsest anchor layer. Therefore, the HAGR approach overcomes the limitation of anchor-graph-based approaches and well compromises classification performance and computational efficiency. The HAGR approach is quite flexible, as we can set different layers of anchors according to the scales of classification tasks. We show that it will degrade to AGR when there is only one anchor layer. In order to further improve the efficiency of the construction of this hierarchical anchor graph, we also investigate an Approximate Nearest Neighbor Search (ANNS) technique to build inter-layer adjacency relationships fastly.

The main contributions of our work are as follows.

1) We make a deep analysis on the existing anchor graph and point out its limitations in dealing with large-scale datasets. That is, AGR faces either an intractable computational cost with dense anchors or a degraded performance with sparse anchors.
2) We propose to build hierarchical anchor graph with a pyramid structure, and develop a scalable classifier based on it. The labels of datapoints are inferred from the coarsest anchors layer by layer and the optimization only involves a small-size reduced Laplacian matrix. The proposed approach overcomes the limitations of AGR and is able to efficiently accomplish large-scale classification with a good performance (detailed computational costs will be shown in Section 4.3).
3) We introduce a fast hierarchical anchor graph construction process, in which the ANNS technique is employed to build inter-layer adjacency relationships.

The rest of this paper is organized as follows. In Section 2, we briefly introduce related work on the graph-based learning. In Section 3, we analyze the traditional AGR approach and its limitations. The proposed approach is described in Section 4. In Section 5, we validate our method and make comparisons with other approaches on large-scale datasets. We also evaluate different graph structures of HAGR to test its flexibility as well as robustness. We finally conclude this paper in Section 6.

2 Related Work

Zhu et al. [50] first introduced the formulation of learning problem based on a Gaussian random field, and analyzed its intimate connections with random walks and spectral graph theory. Zhou et al. [47] subsequently suggested an effective algorithm to obtain the solution of a classification function, which is sufficiently smooth with respect to the intrinsic structure collectively revealed by known labeled and unlabeled datapoints. Later, Zhu et al. [52] developed an improved nonparametric kernel approach by incorporating order constraints during the convex optimization in learning. Zelnik et al. [44] introduced a local scale in computing the affinity between each pair of datapoints for the weighted edge. Meanwhile, inspired by locally linear embedding [31], many works that focus on improving the weight estimation of the graph via sparse representation are proposed. For example, Wang et al. [34] presented a linear neighborhood model for label propagation, which assumes

TABLE 1
Notations and Definitions

Notation	Definition
$\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{E}\}$	An anchor graph or hierarchical anchor graph, where $\mathcal{X}$ and $\mathcal{U}$ indicate datapoints and anchors, respectively, and $\mathcal{E}$ indicates the sets of inter-layer adjacency edges between different sets of points.
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 original data and $\mathcal{L}_b (b \geq 1)$ denotes the bth anchor layer.
m_b	The number of points in $\mathcal{L}_b$.
$\mathbf{Z}^{a,b}$	The inter-layer adjacency matrix between $\mathcal{L}_a$ and $\mathcal{L}_b$. By default, we have $b = a + 1$ for estimating the adjacencies between neighboring layers.
$Z_{is}^{a,b}$	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 used in label smoothness regularization.
Λ^b	The diagonal matrix of the degrees of the anchors in $\mathcal{L}_b$.
$\mathbf{A}$	The soft label matrix of anchors.
$\mathbf{F}$	The soft label matrix of datapoints.
$\mathbf{Y}_{\mathrm{L}}$	The class indicator matrix on labeled datapoints.
$\mathbf{L}$	The reduced Laplacian matrix in the anchor graph or hierarchical anchor graph.
n	The number of datapoints.
c	The number of classes in the dataset.
l	The number of labeled datapoints in the dataset.
$\mathbf{Z}^{\mathrm{H}}$	The accumulated inter-layer adjacency matrix in the hierarchical anchor graph.

that each datapoint can be linearly reconstructed from its neighborhoods with l_2 minimization. Similarly, Cheng et al. [6] proposed a weight estimation method which optimizes the sparse reconstruction coefficients on a l_1 graph. Since selecting local neighbors may lead to disjoint components and incorrect neighbors in graph, Tian et al. [32] advocated learning a nonnegative low-rank graph to capture global linear neighborhoods, under the assumption that each datapoint can be linearly reconstructed from weighted combinations of its direct neighbors and reachable indirect neighbors.

The above graph-based approaches show impressive performances in various applications. However, they are not sufficiently scalable, which imposes limitations in handling larger datasets. With the rapid increase in data size, researchers have paid more attention to designing novel approaches to reduce the computational cost of graph-based learning. Tsang et al. [33] formulated a sparsified manifold regularizer as a center-constrained minimum enclosing ball problem to produce sparse solutions with lower time and space complexities. Wang et al. [35] proposed a multiple random divide-and-conquer approach to construct an approximated neighborhood graph and presented a neighborhood propagation scheme to further enhance the accuracy. Chen et al. [5] presented a method to combine both the original kernel and the graph kernel for scalable manifold regularization.

More recent works seek to employ anchors in scaling up the graph model. Different from the conventional graphs, the anchor-based approaches build the adjacency relationships between original datapoints based on anchors. Zhang et al. [45], [46] first suggested using a set of anchors to perform an effective low-rank approximation of the data manifold, and to span a model suffering the minimum information loss. Liu et al. [23] first presented the anchor graph model, and introduced it into the graph-based learning tasks. Wang et al. [36] subsequently proposed an improved algorithm, which shows better performance and computational efficiency. Compared with the conventional graphs, these anchor-graph-based approaches can largely reduce the complexity in graph construction, and have been widely used in many applications [3], [20], [25], [38]. However, the two-layer anchor graph structure is still limited for processing large-scale learning tasks, which will be analyzed in detail in the following.

3 Anchor-Graph-Based Learning

In this section, we first present a brief description of the anchor-graph-based approach and then give a detailed analysis on its limitations. For convenience, some important notations used throughout the paper and their explanations are listed in Table 1.

3.1 Formulations of AGR

We consider a standard multiclass SSL problem. Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbf{R}^{d \times n}$ with the first l samples being labeled from c distinct classes, anchor-graph-based methods start with clustering a set of representative anchors $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{m_1}\} \in \mathbf{R}^{d \times m_1} (m_1 \leq n)$, which share the same feature space with original datapoints.

To be consistent with the notations in HAGR, here we let $\mathcal{L}_0$ denote the layer of datapoints, and $\mathcal{L}_1$ denote the layer of anchors, as illustrated in Fig. 1a. Different from the conventional graph denoted by an $n \times n$ adjacency matrix, an anchor graph $\mathcal{G}$ is represented by an $n \times m_1$ nonnegative matrix $\mathbf{Z}^{0,1}$, which models inter-layer adjacency relationships between points in $\mathcal{L}_0$ and $\mathcal{L}_1$. Specifically, the entries in each row of $\mathbf{Z}^{0,1}$ are the weights between datapoint $\mathbf{x}_i$ and its k nearest anchors, which can be defined by Nadaraya-Natson kernel regression [14]

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

where the notation $\langle i \rangle \subseteq [1 : m_1]$ is the indices of the k closest anchors of $\mathbf{x}_i$.

Given the labels of anchors and the above inter-layer adjacency, the label of each datapoint can be estimated as a weighted average of them, i.e.,

$$f(\mathbf{x}_i) = \sum_{s=1}^{m_1} Z_{is}^{0,1} f(\mathbf{u}_s), \tag{2}$$

where f is a prediction function that assigns each point a soft label.

Then, to smooth these inferred labels, one can also construct an intra-layer adjacency matrix of datapoints based on inter-layer adjacency relationships

$$\mathbf{W} = \mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}} \in \mathbf{R}^{n\times n}, \tag{3}$$

where the diagonal matrix Λ^1 is defined as $\Lambda^1_{ss} = \sum_{i=1}^{n} Z_{is}^{0,1}$. From Eq. (3), we can see that, $W_{ij} > 0$ means the two datapoints share at least one common anchor, and otherwise $W_{ij} = 0$. It is likely that datapoints sharing common anchors would have similar labels.

Let $\mathbf{Y}_{\mathrm{L}} = [\mathbf{y}_1^{\mathrm{T}}, \ldots, \mathbf{y}_l^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^{l\times c}$ 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. Let $\mathbf{A} = [\mathbf{a}_1{}^{\mathrm{T}}, \mathbf{a}_2{}^{\mathrm{T}}, \ldots, \mathbf{a}_{m_1}{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^{m_1\times c}$ denote the soft label matrix of the anchors in $\mathcal{L}_1$. To deal with the standard multi-class SSL problem, Anchor Graph Regularization [23] is formulated by minimizing $\mathcal{Q}_{\mathbf{A}}$ as

$$\mathcal{Q}_{\mathbf{A}} = \sum_{i=1}^{l} \| \mathbf{Z}_{i\cdot}^{0,1}\mathbf{A} - \mathbf{y}_i \|^2 + \frac{\lambda}{2}\sum_{i,j=1}^{n} W_{ij} \| \mathbf{Z}_{i\cdot}^{0,1}\mathbf{A} - \mathbf{Z}_{j\cdot}^{0,1}\mathbf{A} \|^2, \tag{4}$$

where $\lambda > 0$ is the trade-off parameter balancing different terms, and $\mathbf{Z}_{i\cdot}^{0,1}$ is the ith row of $\mathbf{Z}^{0,1}$. From the above equation, we can see that, the labels of datapoints in both the fitting and smoothness terms are inferred from the anchors. Note that there are other alternative methods for manifold regularization, such as [1], [5]. As it is not the main point of this paper, we simply follow the idea of AGR.

Meanwhile, for the degree of each datapoint, we have $D_{ii} = \sum_j W_{ij} = \sum_{sj} Z_{is}^{0,1}(\Lambda^1_{ss})^{-1} Z_{js}^{0,1} = \sum_s Z_{is}^{0,1} = 1$. Therefore, we obtain the diagonal matrix $\mathbf{D} = \mathbf{I}$, and Eq. (4) is reformulated into a matrix form as

$$\begin{aligned}\mathcal{Q}_{\mathbf{A}} &= \|\mathbf{Z}_{\mathrm{L}}^{0,1}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{0,1\mathrm{T}}(\mathbf{I} - \mathbf{W})\mathbf{Z}^{0,1}\mathbf{A}) \\ &= \|\mathbf{Z}_{\mathrm{L}}^{0,1}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\widetilde{\mathbf{L}}\mathbf{A}),\end{aligned} \tag{5}$$

where $\mathbf{Z}_{\mathrm{L}}^{0,1}$ is the labeled part of $\mathbf{Z}^{0,1}$, and $\widetilde{\mathbf{L}} = \mathbf{Z}^{0,1\mathrm{T}}\mathbf{Z}^{0,1} - (\mathbf{Z}^{0,1\mathrm{T}}\mathbf{Z}^{0,1})(\Lambda^1)^{-1}(\mathbf{Z}^{0,1\mathrm{T}}\mathbf{Z}^{0,1}) \in \mathbf{R}^{m_1\times m_1}$ is the reduced Laplacian matrix in AGR.

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

$$\mathbf{A} = (\mathbf{Z}_{\mathrm{L}}^{0,1\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{0,1} + \lambda\widetilde{\mathbf{L}})^{-1}\mathbf{Z}_{\mathrm{L}}^{0,1\mathrm{T}}\mathbf{Y}_{\mathrm{L}}. \tag{6}$$

Clearly, this matrix inversion takes a time cost of $O(m_1^3)$.

Finally, AGR employs the solved labels associated with the anchors in $\mathcal{L}_1$ to infer the hard label of any unlabeled datapoint in $\mathcal{L}_0$

$$\widehat{y}_i = \mathrm{argmax}_{r\in\{1,\ldots,c\}} \frac{\mathbf{Z}_{i\cdot}^{0,1} \times \mathbf{A}_{\cdot r}}{\beta_r}, i = l+1, \ldots, n, \tag{7}$$

where $\mathbf{A}_{\cdot r}$ is the rth column of $\mathbf{A}$, and $\beta_r = \mathbf{1}^{\mathrm{T}}\mathbf{Z}^{0,1}\mathbf{A}_{\cdot r}$ is the normalization factor, which balances skewed class distributions [50].

3.2 Limitation of AGR: A Dilemma

Compared with the traditional graph models, anchor graph additionally introduces an anchor set into the graph construction. As the number of these anchors can be much smaller than datapoints, both the graph construction and the optimization (especially the inverse computation in Eq. (6)) become much faster. AGR thus becomes a popular tool to handle relatively large datasets.

However, AGR still has limitation in dealing with extremely large-scale datasets. Specifically, it faces a dilemma between performance and computational cost. If we only employ a relatively small number of anchors, the performance of the AGR approaches will degrade as label smoothness regularization and label inference cannot be performed effectively. For the label smoothness regularization, it will introduce many noisy intra-layer edges between dissimilar datapoints by Eq. (3), as they tend to share a distant anchor. For the label inference, it will lead to unreliable integration of label information from k nearest anchors, as inter-layer adjacencies are estimated with too sparse anchors that can be far away from the datapoint. Therefore, to obtain a reasonable accuracy, anchors in the AGR approaches need to be sufficiently dense to build effective adjacency relationships. According to Eq. (6), it results in a dramatically increase of computational cost, which makes the approach practically intractable.

4 Hierarchical Anchor Graph Regularization

We first introduce the definition of hierarchical anchor graph and how we use this graph to build a scalable GSSL approach. Then, we present the efficient graph construction based on ANNS, followed with the analysis on time complexity and other discussions.

4.1 Label Inference and Regularization in HAGR

For graph-based SSL, to obtain good performances in large-scale classification tasks, it always requires an effective smoothness term for regularization, and an efficient solution for model optimization. To build such a scalable graph-based classifier, we extend the anchor graph to a pyramid-like structure and propose a novel graph model called hierarchical anchor graph. For clarity, an illustrative example of the hierarchical anchor graph is shown in Fig. 1b.

Definition 1 (Hierarchical Anchor Graph). *$\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{E}\}$ is a multiple-layer pyramidal graph, where $\mathcal{X}$ indicates the data set, $\mathcal{U}$ indicates the collection of anchor sets, and $\mathcal{E}$ indicates the collection of the adjacency matrices of inter-layer edges between neighboring layers. Suppose the original datapoints $\mathcal{X} \in \mathbf{R}^{d\times n}$ locate in the bottom layer ($\mathcal{L}_0$) of the pyramid. The remaining layers ($\mathcal{L}_b, b = 1, \ldots, h$) are all composed of multiple anchor sets $\mathcal{U}_i$s from fine to coarse, where the size of*

$\mathcal{U}_b \in \mathbf{R}^{d\times m_b}(b=1,\ldots,h)$ is gradually reduced, namely, $m_1 > \cdots > m_h$. All the layers are linked up to a complete graph with h sets of inter-layer adjacency edges, represented by $\mathcal{E}=\{\mathbf{Z}^{0,1},\ldots,\mathbf{Z}^{h-1,h}\}\in\mathbf{R}^{\{n\times m_1,\ldots,m_{h-1}\times m_h\}}$, in which $\mathbf{Z}^{b-1,b}$ denotes the adjacencies between points in $\mathcal{L}_{b-1}$ and $\mathcal{L}_b$.

Based on the hierarchical anchor graph, we can construct a scalable graph-based classifier for multi-class classification tasks, of which the following two key parts are presented in detail: 1) inter-layer adjacency relationships for label inference, which is designed to reduce the number of parameters and make the learning more efficient; and 2) intra-layer adjacency relationships for label smoothness, which is to build an effective regularization and ensure the learning accuracy.

We first pay attention to the former one. Based on the collection of the inter-layer adjacency relationships, we propose to infer the labels of datapoints from $\mathcal{L}_h$ layer by layer throughout the whole graph. As a result, we only need to learn the labels of the coarsest anchors in $\mathcal{L}_h$. Denote $\mathbf{Z}^{\mathrm{H}}$ as the adjacency matrix that estimates the accumulated inter-layer relationships from $\mathcal{L}_0$ to $\mathcal{L}_h$, and we can compute $\mathbf{Z}^{\mathrm{H}}$ as

$$\mathbf{Z}^{\mathrm{H}} = \mathbf{Z}^{0,1}\ldots\mathbf{Z}^{h-1,h}\in\mathbf{R}^{n\times m_h}. \tag{8}$$

Let $\mathbf{A}$ denote the soft label matrix of the anchor set in $\mathcal{L}_h$, and $\mathbf{F}$ denote the inferred label matrix of datapoints in $\mathcal{L}_0$. With the above accumulated matrix, we can conduct the label inference from $\mathcal{L}_h$ to $\mathcal{L}_0$ in a coarse-to-fine manner

$$\begin{aligned}\mathbf{F} &= \mathbf{Z}^{0,1}\mathbf{Z}^{h-1,h}\mathbf{A}\\ &= \mathbf{Z}^{\mathrm{H}}\mathbf{A}.\end{aligned} \tag{9}$$

Next, we consider the label smoothness regularization. In traditional graph-based learning, we prefer sparse intra-layer adjacency matrix because a sparse graph has much less spurious connections between dissimilar points and tends to exhibit high quality. Zhu [49] also pointed out that fully-connected dense graphs perform worse than sparse graphs empirically. Denoting $\mathbf{W}$ as the intra-layer adjacency matrix used in label smoothness regularization, we therefore formulate $\mathbf{W}$ only based on the inter-layer adjacencies between the data layer $\mathcal{L}_0$ and the finest anchor layer $\mathcal{L}_1$ in the hierarchical anchor graph

$$\mathbf{W} = \mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}}\in\mathbf{R}^{n\times n}, \tag{10}$$

where the diagonal matrix Λ^1 is defined as $\Lambda^1_{ss}=\sum_{j=1}^{n} Z^{0,1}_{js}$.

Based on the inferred label matrix $\mathbf{F}$ and the intra-layer adjacency matrix $\mathbf{W}$, we finally obtain Hierarchical Anchor Graph Regularization

$$\mathrm{argmin}_{\mathbf{A}} \parallel \mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A}-\mathbf{Y}_{\mathrm{L}}\parallel^2_{\mathrm{F}} + \frac{\lambda}{2}\sum_{i,j=1}^{n} W_{ij}\parallel \mathbf{Z}^{\mathrm{H}}_{i\cdot}\mathbf{A}-\mathbf{Z}^{\mathrm{H}}_{j\cdot}\mathbf{A}\parallel^2, \tag{11}$$

where $\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}$ is the labeled part of $\mathbf{Z}^{\mathrm{H}}$. Similar to Eq. (4), we have $D_{ii}=\sum_j W_{ij}=1$, and the above expression can be written in the matrix form

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A}-\mathbf{Y}_{\mathrm{L}}\|^2_{\mathrm{F}} + \lambda\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}\mathrm{T}}(\mathbf{I}-\mathbf{W})\mathbf{Z}^{\mathrm{H}}\mathbf{A}),$$

or

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A}-\mathbf{Y}_{\mathrm{L}}\|^2_{\mathrm{F}} + \lambda\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\hat{\mathbf{L}}\mathbf{A}), \tag{12}$$

where $\hat{\mathbf{L}}$ is the reduced Laplacian matrix in HAGR, computed by

$$\begin{aligned}\hat{\mathbf{L}} &= \mathbf{Z}^{\mathrm{H}\mathrm{T}}(\mathbf{I}-\mathbf{W})\mathbf{Z}^{\mathrm{H}}\\ &= \mathbf{Z}^{\mathrm{H}\mathrm{T}}\mathbf{Z}^{\mathrm{H}} - (\mathbf{Z}^{\mathrm{H}\mathrm{T}}\mathbf{Z}^{0,1})(\Lambda^1)^{-1}(\mathbf{Z}^{0,1\mathrm{T}}\mathbf{Z}^{\mathrm{H}})\in\mathbf{R}^{m_h\times m_h}.\end{aligned} \tag{13}$$

As we can see, although our label smoothness regularization is first performed on the labels of all datapoints with the finest anchor layer, the optimization only involves a reduced Laplacian matrix with the size of the coarsest anchor layer. Therefore, HAGR can overcome the limitation of AGR and improve the computation in matrix inversion. Note that since $\mathbf{Z}^{\mathrm{H}}$ is the product of a series of k-sparse adjacency matrices, we will show that the computation of $\hat{\mathbf{L}}$ is also efficient. The detailed computational costs of HAGR will be analyzed later.

With simple derivations, we obtain a global optimal solution for the soft label matrix of the anchor set in $\mathcal{L}_h$ as

$$\mathbf{A} = (\mathbf{Z}^{\mathrm{H}\mathrm{T}}_{\mathrm{L}}\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} + \lambda\hat{\mathbf{L}})^{-1}\mathbf{Z}^{\mathrm{H}\mathrm{T}}_{\mathrm{L}}\mathbf{Y}_{\mathrm{L}}. \tag{14}$$

Based on the learnt labels of the coarsest anchors and the inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$, we can finally infer the hard label for any unlabeled datapoint

$$\widehat{y}_i = \mathrm{argmax}_{r\in\{1,\ldots,c\}}\frac{\mathbf{Z}^{\mathrm{H}}_{i\cdot}\times\mathbf{A}_{\cdot r}}{\beta_r}\pi_r, i=l+1,\ldots,n, \tag{15}$$

where $\beta_r=\mathbf{1}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}\mathbf{A}_{\cdot r}$ is the normalization factor, and π_r is the desirable proportion for class r [50].

From the definition of hierarchical anchor graph, we can see its flexibility. We can vary the number of anchor layers and the number of anchors in each layer. We leave the specific analysis on the parameter settings in the experimental section. In particular, we find that, if our hierarchical anchor graph only contains one anchor layer $(h=1)$, it degrades to the anchor graph, and correspondingly HAGR becomes equivalent to AGR.

4.2 Efficient Graph Construction

Like anchor graph, the construction of a hierarchical anchor graph involves two issues, i.e., the generation of anchor sets and the inter-layer adjacency estimation between neighboring layers.

For the first issue, we can simply follow the anchor graph models in [24], [38] to use a fast clustering algorithms to handle it. As for the issue of the weight estimation, besides the standard kernel regression method, formulating it as a geometric reconstruction problem is an alternative choice [23], [36]. However, the kernel regression takes $O(dnm_1)$ time complexity, while the geometric based methods need more time in solving an optimization problem. For extremely large datasets, both of them can bring intractable computational costs.

To improve the efficiency of the graph construction, we investigate an ANNS technique to accelerate the weight estimation. To obtain adjacency relationships between points in $\mathcal{L}_{b-1}$ and $\mathcal{L}_b$, we first build a *Kmeans tree* $\mathcal{T}$ upon points in $\mathcal{L}_b$. Then, for each point in $\mathcal{L}_{b-1}$, we find its k nearest points

in $\mathcal{L}_b$ with $\mathcal{T}$, and compute a set of l_1 normalized weights between them. With this operation, for example, estimating the adjacencies between n datapoints and m_1 anchors can be efficiently implemented in $O(dn\log m_1)$ [29]. The whole process is summarized in Algorithm 1. Note that aiming at further efficiency, other state-of-the-art techniques, such as Hashing [30], [39], [40], can be considered.

Algorithm 1. *Kmeans-Tree*-Based Inter-Layer Weight Estimation

Input: points in $\mathcal{L}_{b-1}$, points in $\mathcal{L}_b$, number of nearest neighbors k.
1: Employ *Kmeans tree building* algorithm on points in $\mathcal{L}_b$, and obtain a *Kmeans tree* $\mathcal{T}$.
For each point $\mathbf{v}_i$ in $\mathcal{L}_{b-1}$
2: Employ *Kmeans tree searching* algorithm for $\mathbf{v}_i$ on tree $\mathcal{T}$, and obtain indices of its k approximate nearest neighbors $\langle i \rangle$, with the corresponding distances $\mathbf{d}_{\langle i \rangle}$.
3: Compute the l_1 normalized inter-layer weights $\mathbf{z}_{\langle i \rangle} = \exp(-\frac{\mathbf{d}_{\langle i \rangle}}{\sigma})/\hat{z}$, where $\hat{z} = \sum \exp(\frac{-\mathbf{d}_{\langle i \rangle}}{\sigma})$.
End for
4: Construct a sparse inter-layer adjacency matrix $\mathbf{Z}^{b-1,b}$ with the above indices $\langle i \rangle$s and weights $\mathbf{z}_{\langle i \rangle}$s.

Output: $\mathbf{Z}^{b-1,b}$.

Note: The details about *Kmeans tree building* and *searching* algorithms can be found in [29].

4.3 Computational Cost of HAGR

We now analyze the computational cost of HAGR. As inter-layer adjacency matrices in HAGR are all k-sparse, we first introduce the following theorem for the fast sparse matrix multiplication. According to it, for example, the time cost of the accumulated inter-layer matrix $\mathbf{Z}^{\mathrm{H}}$ scales as $O(knm_h)$.

Theorem 1. *Let* $\mathbf{P}$ *and* $\mathbf{Q}$ *be two* $a \times b$ *matrices. If* $\mathbf{Q}$ *contains at most* c *non-zero entries, the naive algorithm can obtain product* $\mathbf{O} = \mathbf{P}\mathbf{Q}^{\mathrm{T}}$ *with* ac *multiplications. The similar bound is obtained when* $\mathbf{P}$ *contains at most* c *non-zero entries. The number of additions required is also bounded by the required number of multiplications.*

The proof of the above theorem can be found in [43].

Then, the steps of HAGR and the corresponding time costs are summarized as follows.

1) Construct a hierarchical anchor graph with Algorithm 1. The computational cost of computing the adjacency matrices is $O(\sum_{b=1}^{h} dm_{b-1}\log m_b)$, where $m_0 = n$. Since practically we usually have $n \gg m_b$, this cost can be approximated as $O(dn\log m_1)$.
2) Calculate the reduced Laplacian matrix $\hat{\mathbf{L}}$ via Eq. (13). As the main cost of this step is the sparse matrix multiplication, based on Theorem 1, the total cost here scales as $O(knm_h)$.
3) Carry out the graph regularization via Eq. (14). The complexity of the matrix inversion is $O(m_h^3)$.
4) Predict the hard labels of unlabeled datapoints via Eq. (15). As we have obtained $\mathbf{Z}^{\mathrm{H}}$ in step 2, it can be conducted efficiently in $O(nm_hc)$.

To sum up, the time complexity of HAGR scales as

$$O(dn\log m_1 + knm_h + m_h^3 + nm_hc),$$

TABLE 2
Comparison of Computational Complexities of Three Graph-Based Methods

Methods	LLGC	AGR	HAGR
Graph construction	$O(dn\log n)$	$O(dn\log m_1)$	$O(dn\log m_1)$
Regularization	$O(n^3)$	$O(knm_1 + m_1^3)$	$O(knm_h + m_h^3)$
Inference	-	$O(nm_1c)$	$O(nm_hc)$

where d is the number of feature dimensions, m_b is the number of anchors in the bth layer, k is the number of nearest neighbors in adjacency estimation, and c is the number of classes.

Here we also summarize the computational costs of Learning with Local and Global Consistency (LLGC [47], a typical graph-based SSL method), AGR and HAGR in Table 2, in which Algorithm 1 is applied into all these methods for a fair comparison. From the table we can observe that, although their complexities in graph construction become linear with respect to the data size and can be comparable, LLGC still has a cubic-time complexity in graph regularization. AGR faces a dramatically increase of computational cost when anchors need to be sufficiently dense for a reasonable accuracy. However, as the scale of the coarsest anchors can be much smaller than the finest anchors, i.e., $m_1 \gg m_h$, HAGR has a much less computational cost and is able to deal with large-scale datasets.

4.4 Discussion on Adjacency Designs

In Section 4.1, we suggest to build the inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$ with anchors layer by layer, and the intra-layer adjacency matrix $\mathbf{W}$ only based on points in $\mathcal{L}_0$ and $\mathcal{L}_1$. Now we present an in-depth analysis on these two aspects.

4.4.1 On the Inter-Layer Adjacency

In HAGR, we model inter-layer adjacency relationships from $\mathcal{L}_0$ to $\mathcal{L}_h$ layer by layer, and then infer the labels of datapoints in a coarse-to-fine manner. According to Eq. (7), it leads to the adaptive relationships between datapoints and the coarsest anchors. That is, when the datapoint is inside the convex envelope of its k nearest anchors in $\mathcal{L}_h$, this datapoint only has connection with these k anchors. When the datapoint is close to the convex envelope's margin, it can build extra inter-layer edges with other nearest anchors in $\mathcal{L}_h$. Otherwise, if we build adjacencies between points in $\mathcal{L}_0$ and $\mathcal{L}_h$ in one step, we are only able to obtain the inflexible relationships between datapoints and their fixed k nearest anchors in $\mathcal{L}_h$.

In the label inference, the above adaptive relationships lead to more reliable integration of label information from the coarsest anchors. Without loss of generality, we demonstrate this by a toy example in Fig. 2, where we have $k = 3$ and $h = 2$. In this example, we want to infer the labels of datapoints $(\mathbf{x}_i, i = 1, 2)$ assisted with the labels of the nearby anchors $(\mathbf{u}_s, s = 1, 2, 3, 4)$ in $\mathcal{L}_2$. In our coarse-to-fine manner, the datapoint $\mathbf{x}_1$, which is inside the convex envelope of its 3 nearest anchors in $\mathcal{L}_2$, receives labels from these 3 anchors. Meanwhile, the datapoint $\mathbf{x}_2$, which is near to a margin of its convex envelope, can receive label information from both $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ and $\mathbf{u}_4$, due to the transitional anchor $\mathbf{u}_4'$

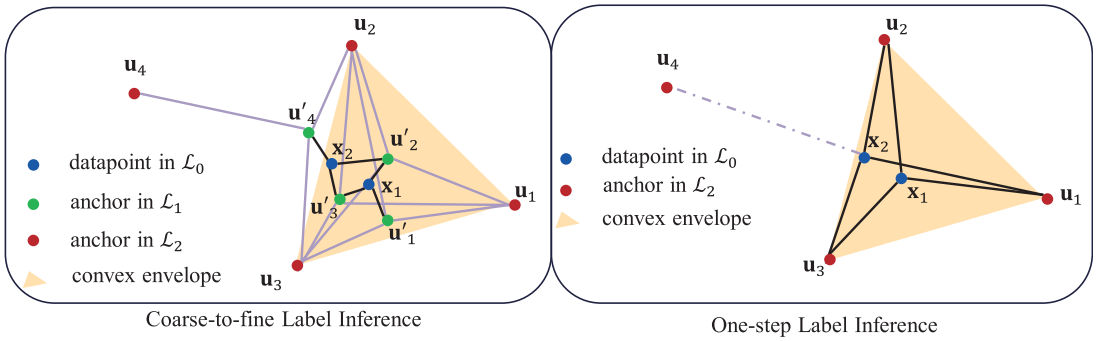


Fig. 2. An illustration of anchor-based label inference in a hierarchical anchor graph. Note that we ignore other points as they have no influence on the label inference here.

in $\mathcal{L}_1$. On the contrary, suppose we conduct the one-step label inference between datapoints and their fixed k coarsest anchors. When $k = 3$, the label information on $\mathbf{u}_4$ is ignored in predicting $\mathbf{y}_2$. When $k = 4$, the noisy labels of the coarsest anchors far away are introduced while inferring $\mathbf{y}_1$. In Section 5.1, we will empirically demonstrate that the classification accuracy can be obviously improved due to this characteristic of $\mathbf{Z}^{\mathrm{H}}$, although the labels of datapoints are still inferred from sparse anchors, namely, the coarsest anchors.

4.4.2 On the Intra-Layer Adjacency

Note that we compute the intra-layer adjacency matrix as $\mathbf{W} = \mathbf{Z}^{0,1}\Lambda_1^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}$. That means, we build $\mathbf{W}$ only based on the anchors in $\mathcal{L}_1$ to force the intra-layer edges stay between similar datapoints. We can also build $\mathbf{W}$ based on anchors in coarser layers, such as using the hth layer, i.e., $\mathbf{W} = \mathbf{Z}^{0,h}\Lambda_h^{-1}\mathbf{Z}^{0,h^{\mathrm{T}}} \in \mathbf{R}^{n\times n}$, $h > 1$, But note that using different anchor layers to build $\mathbf{W}$ leads to nearly the same computational costs of HAGR. Meanwhile, too sparse anchors will arise many incorrect intra-layer edges between dissimilar datapoints since they possibly share a common anchor. That is why we compute $\mathbf{W}$ only using the anchors in $\mathcal{L}_1$.

5 Experiment

In this section, we investigate both the effectiveness and efficiency of our proposed HAGR 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 scales varying from 20,000 to 8,100,000. The descriptions of these datasets are given in below, and some statistics of them are listed in Table 3.

1) *Letter*. The dataset contains 20,000 samples of capital letters from 'A' to 'Z' in the English alphabet [11]. Each sample is converted into 16 primitive numerical attributes (statistical moments and edge counts).
2) *MNIST*. It contains 70,000 samples of handwritten digits from '0' to '9' [21]. 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. We directly use the normalized grayscale value as the feature.
3) *Extended MNIST*. The extended MNIST is widely used in many large-scale graph-based works [15], [19], [24]. The 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 original USPS dataset contains 7,291 training samples of handwritten digits in ten classes [23]. All the digits from '0' to '9' are extended by shifting the 16×16 images in all directions for up to five pixels [33]. There are 882,211 samples in 676 dimensions in total.
5) *MNIST8M*. The MNIST8M dataset has been used in [16], [22], [26] to verify the effectiveness of large-scale learning algorithms. It contains totally 8,100,000 samples in 784 dimensions. In this dataset, the first 70,000 samples belong to the standard MNIST dataset, and each remaining example is generated by applying a pseudo-random transformation to the MNIST training example.

Similar to [23] and [36], the above datasets are categorized into small, medium and large sizes. Specifically, in our experiments, we regard Letter and MNIST as small-size datasets, Extended MNIST and Extended USPS as medium-size datasets, and MNIST8M as a large-size dataset.

5.1 On the Effectiveness of Intra-Layer and Inter-Layer Adjacency Matrices

We conduct experiments on small-size datasets, i.e., Letter and MNIST, to validate the effectiveness of two adjacency designs discussed in Section 4.4.

We first construct a hierarchical anchor graph with two anchor layers, where the size of $\mathcal{L}_1$ is empirically set to

TABLE 3
Details of the Five Databases Used in Our Experiments

	Letter	MNIST	Extended MNIST	Extended USPS	MNIST8M
# of instances	20,000	70,000	630,000	882,211	8,100,000
# of categories	26	10	10	10	10
# of dimensions	16	784	900	676	784

TABLE 4
The Differences of HAGR_{base}, HAGR_W, HAGR_Z, and HAGR in Terms of Label Inference and Label Smoothness Regularization

Approaches	Adjacency Matrix	
	Label Inference Term	Label Smoothness Regularization Term
HAGR_{base}	$\mathbf{Z}^{0,2}$	$\mathbf{Z}^{0,2}(\Lambda^2)^{-1}\mathbf{Z}^{0,2\mathrm{T}}$
HAGR_W	$\mathbf{Z}^{0,2}$	$\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}}$
HAGR_Z	$\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$	$\mathbf{Z}^{0,2}(\Lambda^2)^{-1}\mathbf{Z}^{0,2\mathrm{T}}$
HAGR	$\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$	$\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}}$

$n/10$, and the size of $\mathcal{L}_2$ varies from 100 to 500. For a clear comparison, we change the formulation of AGR to

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}_{\mathrm{L}}\mathbf{A}-\mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2+\frac{\lambda}{2}\sum_{i,j=1}^{n}W_{ij}\|\,\mathbf{Z}_{i\cdot}\mathbf{A}-\mathbf{Z}_{j\cdot}\mathbf{A}\,\|^2, \quad (16)$$

where $\mathbf{W}$ is the intra-layer adjacency matrix for label smoothness regularization and $\mathbf{Z}$ is the inter-layer adjacency for label inference.

Then, based on the above formulation and the hierarchical anchor graph, we define three intermediate versions for HAGR:

1) HAGR_{base}, which has the same structure to AGR and is a baseline for comparison. It only employs anchors in $\mathcal{L}_2$ and datapoints in $\mathcal{L}_0$ to build adjacency matrices for both the label smoothness regularization and label inference.
2) HAGR_W, which is an improved version of HAGR_{base} with a change on label smoothness regularization. This method is compared in order to validate the intra-layer adjacency design. Compared with HAGR_{base}, the only difference is that, for the label smoothness regularization, it builds $\mathbf{W}$ as $\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}}$ in Eq. (16)
3) HAGR_Z, which is an improved version of HAGR_{base} with a change on label inference. This method is compared in order to validate the inter-layer adjacency design with a coarse-to fine manner. Compared with HAGR_{base}, it builds an accumulated inter-layer matrix $\mathbf{Z}$ as $\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$ in Eq. (16).

The differences of HAGR and the above methods are summarized in Table 4. For the other parameters, we set k to 3 to make the graph sparse, and tune λ to its optimal values. In this way, we can provide a fair comparison for these algorithms to validate different adjacency designs. We randomly select 260 and 100 labeled samples for Letter and MNIST, respectively, and leave the remaining ones unlabeled for SSL models.

Table 5 shows the classification accuracies of the above methods. From this table, we have three observations. *First*, by comparing HAGR_W with HAGR_{base}, we can see that, although two methods build the same inter-layer relationships between points in $\mathcal{L}_0$ and $\mathcal{L}_2$, HAGR_W obtains higher accuracies than HAGR_{base}. The main reason is that, by introducing a much sparser intra-layer adjacency matrix, HAGR_W can better smooth the labels of datapoints. *Second*, by comparing HAGR_Z with HAGR_{base}, we can see that, the former obtains better classification performances than the latter, which shows the effectiveness of our adaptive inter-layer adjacency relationships in label inference. *Third*, when the size of $\mathcal{L}_2$ increases, the accuracies of all these approaches increase, and HAGR consistently outperforms the other three methods.

5.2 Comparison with Existing Methods

To demonstrate both the efficiency and effectiveness of the proposed HAGR, we compare it with several state-of-the-art anchor-based SSL models, such as AGR and EAGR. We also report the performance of several baseline methods including 1NN, linear SVM. *For clarity, here we use 'HAGR-m_1-m_2-$\cdots$-m_h' to denote the HAGR method built upon a hierarchical anchor graph with h anchor layers.* For example, 'HAGR-5000-500' means there are two anchor layers in its graph structure, which contain 5,000 and 500 anchors, respectively. The methods for comparison are described in below.

1) The nearest neighbor method, which determines the label of a sample by referring to its closest sample in the labeled set. The method is denoted as '1NN'.
2) Linear SVM [8]. We use the SVM implementation from LIBLINEAR, which is a library for large-scale linear classification. The method is denoted as 'LSVM'.
3) Anchor graph Regularization [23], which is built upon an anchor graph with single anchor layer. It is the prime counterpart in our experiments, and we denote it as 'AGR'.

TABLE 5
Accuracy (%) Comparison of HAGR_{base}, HAGR_W, HAGR_Z, and HAGR on the Letter and MINST Datasets

Dataset	m_1	m_2	HAGR_{base}	HAGR_W	HAGR_Z	HAGR
Letter	2,000	100	45.19 ± 1.53	45.61 ± 1.59	49.19 ± 0.70	$\mathbf{49.57 \pm 1.02}$
($l = 260$)		200	50.91 ± 1.40	51.30 ± 1.31	54.75 ± 1.21	$\mathbf{54.93 \pm 1.05}$
		300	53.72 ± 1.54	54.76 ± 1.73	57.00 ± 1.43	$\mathbf{57.78 \pm 1.48}$
		400	55.58 ± 1.66	56.99 ± 1.57	58.30 ± 1.34	$\mathbf{59.88 \pm 1.42}$
		500	56.80 ± 1.65	58.00 ± 1.69	59.51 ± 1.18	$\mathbf{60.86 \pm 1.36}$
MNIST	7,000	100	79.17 ± 1.39	80.33 ± 1.30	82.33 ± 1.15	$\mathbf{84.59 \pm 0.89}$
($l = 100$)		200	83.28 ± 1.21	83.92 ± 1.17	85.01 ± 0.87	$\mathbf{86.79 \pm 0.66}$
		300	84.44 ± 0.98	85.03 ± 1.02	85.89 ± 0.97	$\mathbf{88.01 \pm 1.00}$
		400	85.30 ± 1.31	85.92 ± 1.26	86.21 ± 1.15	$\mathbf{88.32 \pm 1.15}$
		500	86.35 ± 1.80	86.82 ± 1.74	86.84 ± 1.18	$\mathbf{88.66 \pm 1.23}$

TABLE 6
Classification Accuracies (%) with Different Number of Labeled Samples on the Extended MNIST Dataset

# of labeled samples	1NN	LSVM	AGR	EAGR	HAGR-20,000-5,000	HAGR-200,000-20,000-5,000
100	60.95 ± 0.59	58.58 ± 2.11	88.42 ± 1.36	88.48 ± 1.29	87.97 ± 1.28	$\mathbf{90.75 \pm 1.03}$
200	69.33 ± 0.99	64.07 ± 1.58	90.23 ± 0.44	90.80 ± 0.42	89.81 ± 0.56	$\mathbf{92.21 \pm 0.42}$
300	73.51 ± 0.89	66.99 ± 0.53	91.10 ± 0.38	91.84 ± 0.35	90.65 ± 0.43	$\mathbf{93.13 \pm 0.31}$
400	75.80 ± 0.60	69.50 ± 0.86	91.35 ± 0.34	92.15 ± 0.29	91.15 ± 0.33	$\mathbf{93.39 \pm 0.18}$
500	77.77 ± 0.41	71.47 ± 1.21	92.12 ± 0.18	92.66 ± 0.17	91.96 ± 0.12	$\mathbf{93.73 \pm 0.19}$
600	79.09 ± 0.53	72.69 ± 1.30	92.47 ± 0.10	92.99 ± 0.13	92.27 ± 0.11	$\mathbf{93.95 \pm 0.11}$
700	80.17 ± 0.29	73.69 ± 1.96	92.54 ± 0.11	93.11 ± 0.13	92.43 ± 0.12	$\mathbf{94.05 \pm 0.11}$
800	81.10 ± 0.41	75.22 ± 1.40	92.79 ± 0.13	93.39 ± 0.09	92.61 ± 0.10	$\mathbf{94.15 \pm 0.06}$
900	81.94 ± 0.45	75.93 ± 1.11	93.09 ± 0.09	93.63 ± 0.10	92.93 ± 0.09	$\mathbf{94.23 \pm 0.06}$
1,000	82.60 ± 0.38	76.69 ± 0.95	93.23 ± 0.11	93.77 ± 0.08	93.09 ± 0.10	$\mathbf{94.28 \pm 0.09}$

4) Efficient Anchor graph Regularization [36], which is an improved version of AGR. The method is denoted as 'EAGR'.
5) HAGR-m_1-m_2, which denotes a two-anchor-layer HAGR method. We build the corresponding hierarchical anchor graph by adding a coarser anchor layer above the anchor graph in methods (3-4). The purpose of comparing our approach with this method is to demonstrate the efficiency by introducing a smaller anchor layer.
6) HAGR-m_1-m_2-m_3, which denotes a three-anchor-layer HAGR method. Based on the graph structure in method (5), we add a finer anchor layer with sampled points between its data layer and two anchor layers. This three-anchor-layer HAGR is our proposed approach for the classification on the medium-size and large-size datasets.

Note that here we do not further compare our approach with several other SSL or large-scale classification methods such as conventional graph-based SSL [47], the Eigenfunctions method introduced in [9], the Laplacian SVM method introduced in [28] and the Prototype Vector Machines method introduced in [45], [46], due to the following two facts. First, several methods can hardly be implemented on very large datasets. Second, existing studies have already demonstrated the performance superiority of AGR and EAGR over these methods [23], [36], and thus the superiority of HAGR will be greater if it outperforms AGR and EAGR.

For fair comparisons, we consistently apply the proposed ANNS-based weight estimation algorithm for methods (3-6) in their graph construction, and corresponding kernel widths are set by cross validation. For the above methods, we tune λ to the optimal values.

5.2.1 Medium-Size Datasets

We first conduct experiments on the Extended MNIST and Extended USPS datasets. To accelerate the running speed, we follow [23] and perform PCA to reduce the feature dimension to 86. For the two medium-size datasets, we consistently construct the anchor graph with 20,000 anchors to build AGR and EAGR. Then, by further adding anchor layers, we build two HAGR methods as 'HAGR-20,000-5,000' and 'HAGR-200,000-20,000-5,000', respectively. As for the setting of semi-supervised learning, we vary the number of labeled samples $l = \{100, 200, \ldots, 1,000\}$, while the rest samples remain as unlabeled data.

Averaged over 20 trials, the classification accuracies of the Extend MNIST and Extend USPS datasets are shown in Tables 6 and 7, respectively. The time costs of SSL methods are listed in Table 8.

From these tables, the following observations can be made. *First*, the performances of all graph-based SSL approaches stay at a higher level than Linear SVM and 1NN. This demonstrates the usefulness of unlabeled data in SSL. *Second*, compared with AGR and EAGR, the accuracies of HAGR-20,000-5,000 are slightly lower. However, the performance gap is quite limited-compared with AGR, the accuracy loss of this two-anchor-layer HAGR is smaller than 0.5 percent in most cases. It means that, by building an intra-layer adjacency matrix based on a fixed-size anchor set for label smoothness regularization, the effectiveness of the anchor-based learning can be almost maintained even

TABLE 7
Classification Accuracies (%) with Different Number of Labeled Samples on the Extended USPS Dataset

# of labeled samples	1NN	AGR	EAGR	HAGR-20000-5000	HAGR-200000-20000-5000
100	38.37 ± 0.71	63.82 ± 1.33	64.09 ± 0.85	63.43 ± 1.11	$\mathbf{68.06 \pm 1.55}$
200	47.96 ± 1.14	73.60 ± 0.90	73.75 ± 0.96	73.48 ± 0.96	$\mathbf{77.90 \pm 1.33}$
300	53.48 ± 1.06	78.47 ± 0.91	78.88 ± 0.78	78.28 ± 0.67	$\mathbf{82.28 \pm 0.92}$
400	57.61 ± 1.00	81.41 ± 0.60	81.75 ± 0.45	81.00 ± 0.57	$\mathbf{84.50 \pm 0.81}$
500	61.19 ± 0.80	83.51 ± 0.94	84.09 ± 0.75	84.05 ± 0.78	$\mathbf{86.35 \pm 0.94}$
600	63.94 ± 0.64	84.50 ± 0.88	85.28 ± 0.76	84.09 ± 0.65	$\mathbf{87.11 \pm 0.84}$
700	65.90 ± 0.44	85.80 ± 0.70	86.52 ± 0.56	85.31 ± 0.42	$\mathbf{88.10 \pm 0.49}$
800	67.79 ± 0.38	86.31 ± 0.93	87.59 ± 0.74	86.29 ± 0.69	$\mathbf{88.91 \pm 0.65}$
900	69.18 ± 0.38	86.95 ± 0.68	88.23 ± 0.59	86.95 ± 0.50	$\mathbf{89.44 \pm 0.32}$
1,000	70.71 ± 0.60	87.87 ± 0.55	88.82 ± 0.35	87.55 ± 0.47	$\mathbf{90.01 \pm 0.31}$

TABLE 8
The Comparison of Time Costs (in Seconds) of AGR, EAGR, and HAGR Methods on Medium-Size Datasets

Dataset	AGR	EAGR	HAGR-20,000-5,000	HAGR-200,000-20,000-5,000
Extended MNIST	152.81	151.58	12.12	17.06
Extended USPS	163.31	162.75	17.55	24.18

TABLE 9
Classification Accuracies (%) with Different Number of Labeled Samples on the MNIST8M Dataset

# of labeled samples	1NN	LSVM	AGR	EAGR	HAGR-30,000-5,000	HAGR-300,000-30,000-5,000
100	60.16 ± 1.96	59.67 ± 2.19	89.87 ± 1.78	90.27 ± 0.18	89.46 ± 1.24	$\mathbf{91.36 \pm 0.70}$
200	68.66 ± 1.29	64.46 ± 2.37	91.15 ± 0.59	91.76 ± 0.57	90.85 ± 0.50	$\mathbf{92.46 \pm 0.42}$
300	72.78 ± 0.81	66.79 ± 2.25	92.21 ± 0.51	92.37 ± 0.51	91.66 ± 0.42	$\mathbf{93.05 \pm 0.37}$
400	75.33 ± 0.60	68.33 ± 1.97	92.47 ± 0.44	92.73 ± 0.38	92.16 ± 0.36	$\mathbf{93.43 \pm 0.37}$
500	77.24 ± 0.55	70.65 ± 1.49	92.70 ± 0.41	93.05 ± 0.29	92.50 ± 0.29	$\mathbf{93.78 \pm 0.24}$
600	78.58 ± 0.54	72.64 ± 1.36	92.80 ± 0.34	93.17 ± 0.27	92.64 ± 0.26	$\mathbf{93.90 \pm 0.27}$
700	79.87 ± 0.70	73.80 ± 1.27	93.12 ± 0.31	93.41 ± 0.30	92.92 ± 0.28	$\mathbf{94.10 \pm 0.25}$
800	81.02 ± 0.50	73.87 ± 1.18	93.19 ± 0.23	93.51 ± 0.15	93.06 ± 0.16	$\mathbf{94.21 \pm 0.15}$
900	81.76 ± 0.49	73.97 ± 0.96	93.29 ± 0.36	93.63 ± 0.21	93.18 ± 0.26	$\mathbf{94.28 \pm 0.16}$
1,000	82.51 ± 0.42	76.95 ± 1.13	93.49 ± 0.22	93.79 ± 0.15	93.37 ± 0.16	$\mathbf{94.39 \pm 0.12}$

TABLE 10
The Comparison of Time Costs (in Seconds) of AGR, EAGR, and HAGR Methods on the MNIST8M Dataset

Dataset	AGR	EAGR	HAGR-30,000-5,000	HAGR-300,000-30,000-5,000
MNIST8M	665.07	662.60	104.97	137.54

we significantly reduce the size of to-be-learned anchor set (from 20,000 to 5,000). When taking into account the running time of the learning process (shown in Table 8), this accuracy loss becomes acceptable in real applications. *Third*, by increasing the size of $\mathcal{L}_1$ over 20,000, the performances of the anchor-based approaches can be further improved due to better adjacency relationships. Based on this, HAGR-200,000-20,000-5,000 consistently outperforms AGR and EAGR. When the number of labeled samples is small, this advantage is more obvious, e.g., improvements of about 2 and 4 percent on Extended MNIST and Extended USPS, respectively. It verifies the effectiveness of HAGR by introducing a finer anchor layer to improve the graph regularization. *Fourth*, although the ANNS-based graph construction can be applied to all graph-based approaches, it is particularly suitable for our HAGR. Due to this efficient graph construction and the fast optimization, HAGRs overcome the limitation of AGR and EAGR, and achieve good performances with less computational costs.

5.2.2 Large-Size Dataset

To demonstrate the scalability of HAGR, we conduct experiments on the MNIST8M dataset, where the dimension of examples is also reduce to 86 by PCA. We first construct an anchor graph with 30,000 anchors to build AGR and EAGR. Then, we build two HAGR methods, i.e., 'HAGR-30000-5,000' and 'HAGR-300,000-30,000-5,000', respectively. By repeating the similar evaluation process, we display the classification accuracies over 20 trials in Table 9. The time costs of SSL approaches are listed in Table 10.

From these tables, we have the following observations. *First*, compared with AGR and EAGR, the accuracy loss of HAGR-30,000-5,000 is acceptable while its time cost is much less. *Second*, as the number of labeled data varies from 100 to 1,000, the performances of all methods increase, and our HAGR-300,000-30,000-5,000 consistently outperforms the other methods.

It is also worth noting that, in our experimental results, we actually have put more emphasis on the performance superiority of the three-anchor-layer HAGR. One may argue that we can increase the number of anchors in AGR and EAGR, say, setting 300,000 anchors. But this will dramatically increase the time costs of these methods (increased by about 10^3 times), which are practically intractable.

5.3 On the Structure of Hierarchical Anchor Graph

We can see that HAGR is a quite flexible approach, and thus designing the structure of the hierarchical anchor graph can be important in classification tasks. Therefore, we conduct additional experiments by varying the number of anchor layers (h) and the size of each anchor layer (m_b) in the proposed graph model, trying to investigate the impact of these parameters. For convenience, Extended USPS dataset is used in this experiment. We build three anchor graphs with 5,000, 10,000, and 20,000 anchors for implementing AGR and EAGR as comparisons.

For clarity, in the pictures illustrating results, we use black/blue/red lines to show the results of HAGR built with 2/3/4 anchor layers, respectively. Among them, each dot-curve denotes HAGRs with the varying anchor size m_b, and the straight line without dots means the size of each anchor layer of this HAGR is fixed. The specific size is displayed at the bottom of each subfigure.

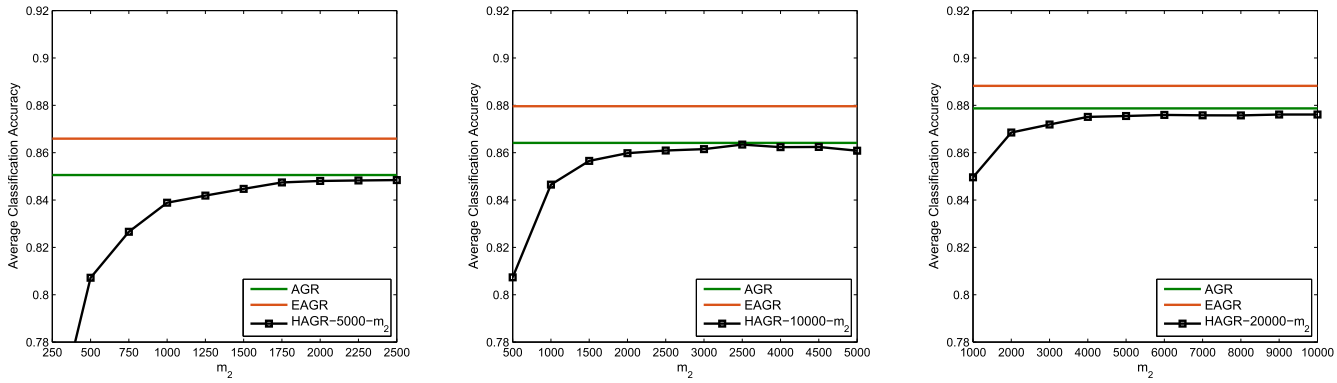


Fig. 3. Average performance curves with respect to the variation of m_2 in HAGR-m_1-m_2.

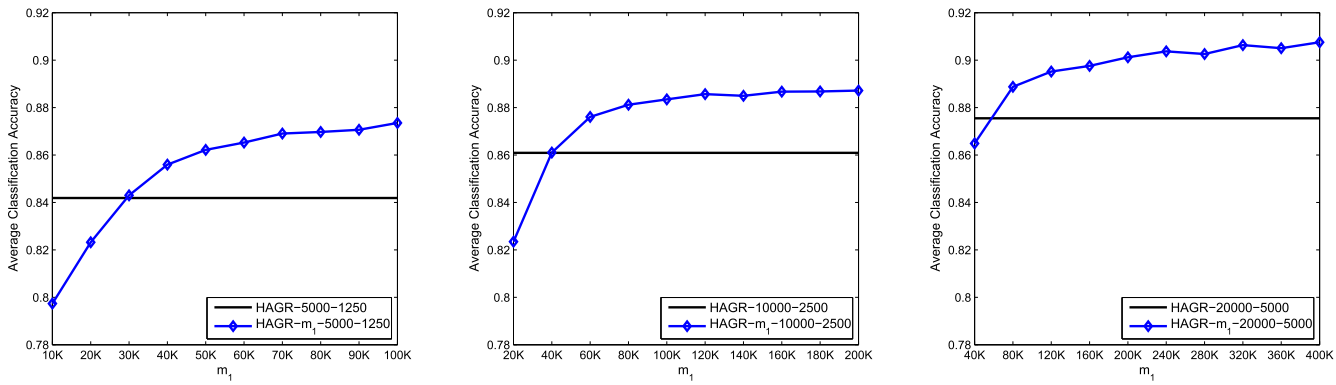


Fig. 4. Average performance curves with respect to the variation of m_1 in HAGR-m_1-m_2-m_3.

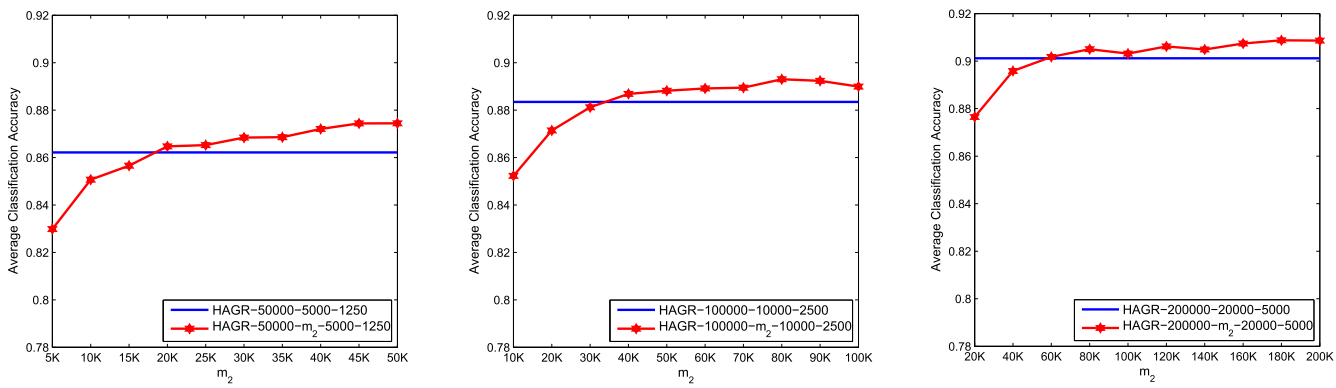


Fig. 5. Average performance curves with respect to the variation of m_2 in HAGR-m_1-m_2-m_3-m_4.

We first test the two-anchor-layer HAGR method. Similar to the process in Section 5.2, for each of the three anchor graphs, we add a coarser anchor layer $\mathcal{L}_2$ above the original $\mathcal{L}_1$ to construct our hierarchical anchor graph ($h = 2$). The performance curves of HAGR-m_1-m_2 with respect to the size of $\mathcal{L}_2$ are shown in Fig. 3. As we can see, the accuracy of HAGR increases rapidly at the first stage. It means that, although we can build a large-scale finest anchor layer in HAGR to improve the performance, the size of to-be-learned anchors cannot be too small. Otherwise, it tends to restrict the performance of HAGR. When m_2 reaches about $m_1/4$, the accuracy of the HAGR becomes stable. We note that this number of to-be-learned anchors m_h can still be much smaller than the number of anchors m_1 used in label smoothness regularization.

Then, we investigate the three-anchor-layer HAGR method. Based on each two-anchor-layer graph above, we add a finer anchor layer $\mathcal{L}_1$ with m_1 anchors between the original data layer and two anchor layers to construct new graph ($h = 3$). The number of the coarsest anchors is set to the empirical value, and the specific settings can be found in Fig. 4. The performance curve of HAGR-m_1-m_2-m_3 with respect to m_1 is shown in Fig. 4. As we can see, larger m_1 brings higher accuracy. The reason is that, by increasing the size of the finest anchor layer, the representation power of these anchors becomes stronger. Then, we can obtain more

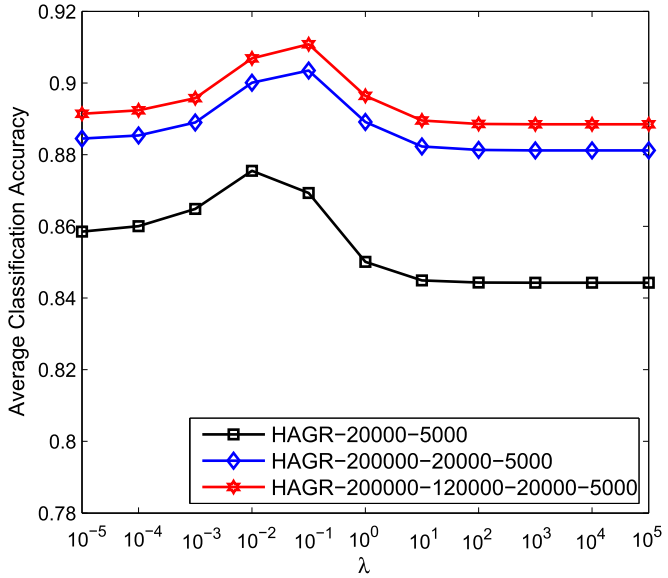


Fig. 6. Average performance curves of HAGR with respect to the variation of λ.

effective adjacency relationships for both the label smoothness regularization and label inference.

We also build the four-anchor-layer HAGR method ($h = 4$) by further adding an anchor layer between original $\mathcal{L}_1$ and $\mathcal{L}_2$ in each three-anchor-layer graph above. The performance curve of HAGR-m_1-m_2-m_3-m_4 with respect to current m_2 is shown in Fig. 5. As we can see, further increasing the number of anchor layers can improve the accuracy, which demonstrates the effectiveness of exploring multiple-layer anchors to model the coarse-to-fine label inference.

To summarize, when fixing the number of anchor layers in hierarchical anchor graph, the performance of HAGR is fairly robust to the number of anchors in each layer over a large range (as we can see in each subfigure), and the to-be-learned anchors can be much fewer than the anchors used in label smoothness regularization. Meanwhile, increasing the number of anchor layers can improve accuracy (under a comparison from Figs. 3 to 5), and in our experiments, we empirically find that three or four anchor layers are already able to well handle million-scale datasets.

5.4 On the Trade-Off Parameter λ

We also test the sensitivity of parameter λ in the proposed approach. For simplicity, we only illustrate the results of HAGR with $m_h = 5,000$ on the Extended USPS dataset. But similar observations can also be obtained in other cases. As we can see in Fig. 6, the performance of HAGR will not severely degrade when λ varies in a wide range, and increasing the number of anchor layers does not change the robustness of λ.

6 Conclusion

This work proposes a novel Hierarchical Anchor Graph Regularization approach by exploring multiple-layer anchors in a pyramid-style structure. It generalizes the conventional graph-based and anchor-graph-based SSL methods to a hierarchical approach. In HAGR, we perform label smoothness regularization on all datapoints based on the finest anchors. By inferring the labels of datapoints started from the coarsest anchors, we obtain an efficient optimization which only involves a small-size reduced Laplacian matrix. It overcomes the limitations of existing AGR approach in dealing with extremely large datasets. We also investigate ANNS to improve the efficiency of the construction of the hierarchical anchor graph. Experiments on publicly available large-scale datasets of various sizes have demonstrated the effectiveness of our approach in terms of computational speed and classification accuracy.

Acknowledgments

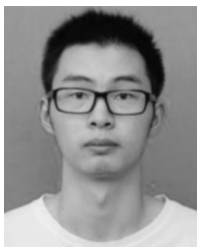
This work is partially supported by the National 973 Program of China under grants 2014CB347600 and 2013CB329604, the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059, and the National Nature Science Foundation of China under grant 61432019.

References

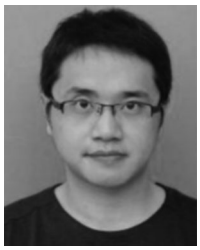
[1] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 11, pp. 2399–2434, 2006.
[2] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.
[3] D. Cai and X. Chen, "Large scale spectral clustering with landmark-based representation," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug. 2015.
[4] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recog. Lett.*, vol. 16, no. 1, pp. 105–111, 1995.
[5] L. Chen, I. W. Tsang, and D. Xu, "Laplacian embedded regression for scalable manifold regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 902–915, Jun. 2012.
[6] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang, "Learning with $l1$-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.
[7] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao, "Visual reranking through weakly supervised multi-graph learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2600–2607.
[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, no. 9, pp. 1871–1874, 2008.
[9] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Proc. Conf. Advances Neural Inf. Proc. Syst.*, 2009, pp. 522–530.
[10] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *Int. J. Pattern Recog. Artif. Intell.*, vol. 28, no. 1, pp. 178–215, 2014.
[11] P. W. Frey and D. J. Slate, "Letter recognition using holland-style adaptive classifiers," *Mach. Learn.*, vol. 6, no. 2, pp. 161–182, 1991.
[12] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. T. Shen, "Optimal graph learning with partial tags and multiple features for image and video annotation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 4371–4379.
[13] M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 902–909.
[14] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer, 2010.
[15] M. Hein and S. Setzer, "Beyond spectral clustering-tight relaxations of balanced graph cuts," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2011, pp. 2366–2374.
[16] C.-J. Hsieh, S. Si, and I. S. Dhillon, "A divide-and-conquer solver for kernel support vector machines." in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 566–574.
[17] S. Huang, M. Elhoseiny, A. Elgammal, and D. Yang, "Learning hypergraph-regularized attribute predictors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 409–417.

[18] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.
[19] M. Karlen, J. Weston, A. Erkan, and R. Collobert, "Large scale manifold transduction," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 448–455.
[20] S. Kim and S. Choi, "Multi-view anchor graph hashing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 3123–3127.
[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[22] M. Li, J. T.-Y. Kwok, and B. Lü, "Making large-scale nyström approximation possible," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 631–638.
[23] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 679–686.
[24] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proc. IEEE*, vol. 100, no. 9, pp. 2624–2638, Sep. 2012.
[25] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.
[26] L. Bottou, "Large-scale kernel machines," MIT press, 2007.
[27] D. Machin, *Introduction to Multimodal Analysis*. London, U.K.: Bloomsbury Publishing, 2016.
[28] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, no. 5, pp. 1149–1184, 2009.
[29] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.
[30] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.
[31] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[32] Z. Tian and R. Kuang, "Global linear neighborhoods for efficient label propagation," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 863–872.
[33] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 363–392, 2005.
[34] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[35] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable *k*-NN graph construction for visual descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 1106–1113.
[36] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, Jul. 2016.
[37] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.
[38] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He, "EMR: A scalable graph-based ranking model for content-based image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 102–114, Jan. 2015.
[39] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, and H. T. Shen, "Zero-shot hashing via transferring supervised knowledge," in *Proc. ACM Conf. Multimedia*, 2016, pp. 1286–1295.
[40] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li, "Robust discrete spectral hashing for large-scale image semantic indexing," *IEEE Trans. Big Data*, vol. 1, no. 4, pp. 162–171, Dec. 2015.
[41] Y. Yang, Y. Yang, H. T. Shen, Y. Zhang, X. Du, and X. Zhou, "Discriminative nonnegative spectral clustering with out-of-sample extension," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1760–1771, Aug. 2013.
[42] G. Yu, G. Zhang, Z. Zhang, Z. Yu, and L. Deng, "Semi-supervised classification based on subspace sparse representation," *Knowl. Inf. Syst.*, vol. 43, no. 1, pp. 81–101, 2015.
[43] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.
[44] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2005, pp. 1601–1608.
[45] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 1233–1240.
[46] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semi-supervised learning via prototype vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 444–457, Mar. 2015.
[47] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2004, pp. 321–328.
[48] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2006, pp. 1601–1608.
[49] X. Zhu, "Semi-supervised learning literature survey," Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2005.
[50] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.
[51] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.
[52] X. Zhu, J. Kandola, Z. Ghahramani, and J. D. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2004, pp. 1641–1648.

Meng Wang received the BE and PhD degrees from the Special Class for the Gifted Young, Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. He is a professor with the Hefei University of Technology, China. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, and journal and conference papers in these areas. He received the ACM SIGMM Rising Star Award 2014. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE Transactions on Circuits and Systems for Video Technology*. He is a member of the IEEE.



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.



Shijie Hao received the BE, MS, and PhD degrees from the School of Computer and Information, Hefei University of Technology (HFUT). He is an associate professor with HFUT, China. His current research interests include machine learning and image processing.

Hengchang Liu received the PhD degree from the University of Virginia, in 2011, under the supervision of Professor John Stankovic. He is currently an assistant professor with USTC. His research interests mainly includes big data mining, cyber physical systems, and mobile systems.

Xindong Wu received the bachelor's and master's degrees in computer science from the Hefei University of Technology, China, and the PhD degree in artificial intelligence from the University of Edinburgh, Britain. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and a professor of computer science with the University of Louisiana at Lafayette. His research interests include data mining, knowledge-based systems, and Web information exploration. He is the steering committee chair of the IEEE International Conference on Data Mining (ICDM), the editor-in-chief of *Knowledge and Information Systems* (*KAIS*, by Springer), and a series editor of the *Springer Book Series on Advanced Information and Knowledge Processing* (*AI&KP*). He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*, by the IEEE Computer Society) between 2005 and 2008. He served as program committee chair/co-chair for ICDM '03 (the 2003 IEEE International Conference on Data Mining), KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), and CIKM 2010 (the 19th ACM Conference on Information and Knowledge Management). He is a fellow of the IEEE.