Applied Intelligence
https://doi.org/10.1007/s10489-022-04333-2

Multiscale laplacian learning

Ekaterina Merkurjev[1] · **Duc Duy Nguyen**[2] · **Guo-Wei Wei**[3]

Accepted: 8 November 2022

Abstract
Machine learning has greatly influenced a variety of fields, including science. However, despite tremendous accomplishments of machine learning, one of the key limitations of most existing machine learning approaches is their reliance on large labeled sets, and thus, data with limited labeled samples remains an important challenge. Moreover, the performance of machine learning methods is often severely hindered in case of diverse data, which is usually associated with smaller data sets or data associated with areas of study where the size of the data sets is constrained by high experimental cost and/or ethics. These challenges call for innovative strategies for dealing with these types of data.
In this work, the aforementioned challenges are addressed by integrating graph-based frameworks, semi-supervised techniques, multiscale structures, and modified and adapted optimization procedures. This results in two innovative multiscale Laplacian learning (MLL) approaches for machine learning tasks, such as data classification, and for tackling data with limited samples, diverse data, and small data sets. The first approach, multikernel manifold learning (MML), integrates manifold learning with multikernel information and incorporates a warped kernel regularizer using multiscale graph Laplacians. The second approach, the multiscale MBO (MMBO) method, introduces multiscale Laplacians to the modification of the famous classical Merriman-Bence-Osher (MBO) scheme, and makes use of fast solvers. We demonstrate the performance of our algorithms experimentally on a variety of benchmark data sets, and compare them favorably to the state-of-art approaches.

Keywords Graph-based methods · Manifold learning · Multiscale framework · Graph laplacian

1 Introduction

Artificial intelligence, including machine learning, has irreversibly changed many fields including science, engineering, and technology [51, 56]. In fact, the combination of artificial intelligence (AI) and big data has been referred to as the "fourth industrial revolution" [95]. Nevertheless, machine learning approaches face several challenges.

First, while the big data challenge is well known, little attention is paid to the diverse data challenge. The success behind machine learning is that the behavior in unknown domains can be accurately estimated by quantitatively learning the pattern from sufficient training samples. However, while data sets in computer vision and image analysis often contain millions or billions of points, it is typically difficult to obtain large data sets in experimental science [49]. We often deal with diverse data originating from a relatively small data set lying in a huge space. For example, due to the complexity, ethnics, and high cost of scientific experiments [45, 92, 96, 97], it is extremely difficult to collect a relatively small set of drug candidates of the order of 10^6 for a therapeutic target, while the size of the underlying chemical space of potentially pharmacologically active molecules is about 10^{60} [11]. Therefore, researchers try to cover as many components as possible with a small number of sampling points. The diversity

✉ Ekaterina Merkurjev
merkurje@msu.edu

Duc Duy Nguyen
ducnguyen@uky.edu

Guo-Wei Wei
weig@msu.edu

[1] Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA

[2] Department of Mathematics, University of Kentucky, Lexington, KY 40506, USA

[3] Department of Mathematics, Department of Biochemistry and Molecular Biology, Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, USA

Published online: 28 November 2022

is created by deliberately sampling a wide distribution in the huge space to understand the landscape of potential drugs. This practice is very common in scientific explorations. Similar diverse data sets exist in materials design [32, 116]. Overall, diverse data originated from a relatively small data set lying in a huge chemical space gives rise to a severe challenge for machine learning. Mathematically, diverse data involves disconnected submanifolds and/or nested submanifolds corresponding to multiphysics and multiscale natures of the diversity, respectively [25, 74]. The multiphysics and multiscale representations of data have been addressed by the authors' earlier work on element-specific persistent homology [16, 18–20]. However, multiscale graph learning models have hardly been developed. The proposed algorithms of this paper fill the gap, addressing the multiphysics nature of data diversity through a multiphysics data representation, such as the element-specific feature extraction developed in recent works such as [16, 18–20, 73].

Second, the success of many existing approaches for machine learning tasks, such as data classification, is dependent on a sufficient amount of labeled samples. However, obtaining enough labeled data is difficult as it is time-consuming and expensive, especially in domains where only experts can determine experimental labels; thus, labeled data is scarce. As a result, the majority of the data embedded into a graph is unlabeled data, which is often much easier to obtain than labeled data but more challenging to predict. Overall, one of the key limitations of most existing approaches is their reliance on large labeled sets; in particular, deep learning approaches often require massive labeled sets to learn the patterns behind the data. These challenges call for innovative strategies to revolutionize the current state-of-the-art.

Recently, algorithms involving the graph-based framework, such as those described in Section 2.1, have recently become some of the most competitive approaches for applications ranging from image processing to social sciences. Such methods have been successful in part due to the many advantages offered by using a graph-based approach. For example, a graph-based framework provides valuable information about the extent of similarity between elements of both labeled and unlabeled data via a weighted similarity graph and also yields information about the overall structure of the data. Moreover, in addition to handling nonlinear structure, a graph setting embeds the dimension of the features in a graph during weight computations, thus reducing the high-dimensionality of the problem. The graph framework is also able to incorporate diverse types of data, such as 3D point clouds, hyperspectral data, text, etc.

Inspired by the recent successes, we address the aforementioned challenges by integrating similarity graph-based frameworks, multiscale structure, modified and adapted optimization techniques and semi-supervised procedures, with both labeled and unlabeled data embedded into a graph. Overall, this paper formulates two multiscale Laplacian learning (MLL) approaches for machine learning tasks, such as data classification, and for dealing with diverse data, data with limited samples and smaller data sets. The first approach, the multikernel manifold learning (MML) method, introduces multiscale kernels to manifold regularization. This approach integrates new multiscale graph Laplacians into loss-function based minimization problems involving warped kernel regularizers. The second approach, the multiscale Merriman-Bence-Osher (MMBO) method, adapts and generalizes the classical Merriman-Bence-Osher (MBO) scheme [71] to a multiscale graph Laplacian setting for learning tasks. The MMBO approach also makes use of fast solvers, such as [6, 10, 33, 34], for finding approximations of the extremal eigenvectors of the graph Laplacian. We validate the proposed MLL approaches using a variety of data sets.

There are several strengths of the proposed methods:

- The proposed techniques address the multiscale nature of data through a multiphysics data representation, allowing them to perform well in the case of diverse data, which often occurs in, e.g., scientific applications.
- The methods require less labeled training data to accurately classify a data set compared to most existing machine learning techniques, especially supervised approaches, and often in considerably smaller quantities. This is in part due to the usage of a similarity graph-based framework and the fact that the majority of the data embedded into the graph is unlabeled data. In fact, in most cases, good accuracy can be obtained with *at most* 1%-5% of the data elements serving as labeled data. This is an important advantage due to the scarcity of labeled data for most applications.
- Although equally applicable and successful in the case of larger data, the new methods also perform well in the case of smaller data sets, which often result in unsatisfactory performances for existing machine learning techniques, due to an often insufficient number of labeled samples and a decreased ability for machine learning-based models to learn from the observed data.

The proposed MMBO method offers specific advantages:

- Although it can perform just as successfully on smaller data, the MMBO algorithm is equipped with a structure which allows it to be easily adapted and designed specifically for the use of large data. In particular, in the case of large data, one can use a slight modification of the fast Nyström extension procedure [10, 33, 34] to compute an approximation to the extremal eigenvectors of the multiscale graph Laplacian using a dense graph

without the need to compute all the graph weights. In fact, only a small portion of the weights need to be calculated. Overall, the method uses a low-dimensional subspace spanned by only a small number of eigenfunctions.
– Once the N_e eigenvectors of the graph Laplacian are computed, the complexity of this algorithm is linear. Moreover, the Nyström extenstion procedure allows the N_e eigenvectors of the graph Laplacian to be computed using only $O(NN_e)$ operations, where $N_e << N$ and N is the number of data elements.

The paper is organized as follows. In Section 2, we present background, previous work and an overview of graph learning methods. In Section 3, we derive the proposed MML and MMBO methods and provide details on the computation of eigenvectors of the graph Laplacian for the latter method. The results from experiments are described in Section 4, and we present a conclusion in Section 5.

2 Background

2.1 Previous work

In this section, we review recent graph-based methods for data classification and semi-supervised learning, including approaches related to convolutional neural networks, support vector machines, neural networks, label propagation, embedding methods, multi-view and multi-modal methods.

Convolutional neural networks have recently been extended to a graph-based framework for the purpose of semi-supervised learning. In particular, [55] presents a scalable approach using graph convolutional networks by integrating a convolutional architecture motivated by a localized first-order approximation of spectral graph convolutions. Deeper insights into the graph convolutional neural network model are discussed in [58]. Moreover, a dual graph-based convolutional network approach is described in [123], while a Bayesian graph convolutional network procedure is derived in [117]. In [4], a multi-scale graph convolution model is presented. In [13], generalizations of convolutional neural networks to signals defined on more general domains using two constructions are described; one of the generalizations is based on the spectrum of the graph Laplacian.

Neural networks have also been extended to a graph-based framework for the task of semi-supervised learning. For example, attention-based graph neural networks [102], graph partition neural networks [60], and graph Markov neural networks [90] have been proposed.

Moreover, support vector machines are also applied to semi-supervised learning using a graph-based framework. In [23], graph-based support vector machines are derived to emphasize low density regions. Also, Laplacian support vector machines (LapSVM) [9, 64] and Laplacian twin support vector machines (Lap-TSVM) [88] have been formulated.

Label and measure propagation methods are discussed in, e.g., [46], where the authors derive a transductive label propagation method that is based on the manifold assumption. Label propagation techniques and the use of unlabeled data in classification are investigated in [121]. Dynamic label propagation is studied in [104], while semi-supervised learning with measure propagation is described in [100].

Embedding methods are also used for semi-supervised learning. Nonlinear embedding algorithms for use with shallow semi-supervised learning techniques, such as kernel methods, are applied to deep multi-layer architectures in [110]. Other graph embedding methods are presented in [114].

Multi-view and multi-modal methods include [81], which proposes a reformulation of a standard spectral learning model that can be used for multiview clustering and semi-supervised tasks. The work [80] proposes novel multi-view learning, while [40] describes multi-modal curriculum learning.

Other techniques for graph-based semi-supervised learning include fast anchor graph regularization [106], a Bayesian framework for learning hyperparameters [52], and random subspace dimensionality reduction. In [39], a classification method is proposed to learn from dissimilarity and similarity information on labeled and unlabeled data using a novel graph-based encoding of dissimilarity. Random graph walks are used in [61], and sampling theory for graph signals is utilized in [36]. In [105], a bivariate formulation for graph-based semi-supervised learning is shown to be equivalent to a linearly constrained max-cut problem. Lastly, reproducing kernel Hilbert spaces are used in [99].

Various approaches involving graph-based regularization terms include regularization frameworks [119, 120], regularization developments [22], anchor graph regularization [106], manifold regularization [9], measure propagation [100], approximate energy minimization [12], nonlocal discrete regularization [30], power watershed [29], spectral matting [57], Laplacian regularized least squares [99], locality and similarity preserving embedding [31], and clustering [83]. Examples for graph Laplacian regularization include label propagation [121] and deep semi-supervised embedding [110].

Merkurjev and co-authors have studied graph-based spectral approaches [37, 38, 65–70] using Ginzburg-Landau techniques and modifications of the MBO scheme [71], which is an efficient method for evolving an interface by mean curvature in a continuous setting and which can be linked to optimization problems involving the Ginzburg-Landau functional. Specifically, the MBO scheme can be

derived from a Ginzburg-Landau functional minimization procedure, and can be modified and transferred to a graph setting using more general operators on graphs, as shown in Merkurjev's work on data classification [37, 65, 68–70].

Overall, Merkurjev and co-authors have shown that multiclass data classification can be achieved using techniques from topological spaces and the Gibbs simplex [37, 68]. In particular, MBO-like methods were developed for image processing applications [69], hyperspectral imaging [38, 70], Cheeger and ratio cut applications [67], heat kernel pagerank applications [66], and unsupervised learning [65]. The subject of this paper is to integrate elements of this prior work, prior work on manifold learning and novel graph-based formulations into a multiscale framework to develop new multiscale graph-based methods for machine learning tasks, such as data classification. Our methods will be able to deal with a variety of scales present in many data sets.

Finally, two related methods involving dimension reduction [63, 107]. The first method proposes an adaptive discriminative analysis framework for embedding non-Gaussian data and can preserve within-class local structure and learn discriminative transformation functions simultaneously. The second method [107] introduces novel convolutional two-dimensional nonlinear discriminant analysis for dimensionality reduction and utilizes the nonlinearity of the CNN. This method benefits from its learning ability.

2.2 Graph-based framework

The methods presented in this work use a similarity graph framework consisting of a graph $G = (V, E)$, where $V = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is a set of vertices associated with the elements of the data set, and E is a set of edges connecting some pairs of vertices. The edges are weighted by a weight function $w : V \times V \to \mathbb{R}$, where $w(\mathbf{x}_i, \mathbf{x}_j)$ measures the degree of similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Larger values indicate similar elements and smaller values indicate dissimilar elements. Naturally, the embedding of data into a graph depends greatly on the edge weights. This section provides more details about graph construction, but the exact manner of weight construction for particular data sets is described in Section 4.

The use of the graph-based framework offers many advantages. First, it provides valuable information about the extent of similarity between pairs of elements of both labeled and unlabeled data via a weighted similarity graph and also yields information about the overall structure of the data. This reduces the amount of labeled data needed for good accuracy. Moreover, a graph-based setting embeds the dimension of the features in the graph during weight computations, thus reducing the high-dimensionality of the problem. It also provides a way to handle nonlinearly separable classes and affords the flexibility to incorporate diverse types of data. In addition, in image processing, the graph setting allows one to capture texture more accurately.

The exact technique of computing the similarity value between two elements of data depends on the data set, but first involves feature (attribute) vector construction and a distance metric chosen specifically for the data and task at hand. For example, for hyperspectral data, one may choose the feature vector to be the vector of intensity values in its many bands and the distance measure to be the cosine distance. For 3D sensory data, one can take the feature vector to contain both geometric and color information; the weights can be calculated using a Gaussian function incorporating normal vectors, e.g., [7]. For text classification, popular feature extraction methods include term frequency- inverse document frequency and bag-of-words, both described in [2]. For biological data tasks, such as protein classification, persistent homology [16] can be used for feature construction.

Once the features are constructed, the weights are computed. Popular weight functions include the Zelnik-Manor and Perona function [87] and the Gaussian function [103]:

$$w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma^2}\right), \tag{1}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ represents a distance between feature vectors of data elements $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\sigma > 0$. Using the weight function w, one can construct a weight matrix $\mathbf{W}$ defined as $\mathbf{W}_{ij} = w(\mathbf{x}_i, \mathbf{x}_j)$, and define the degree of a vertex $\mathbf{x}_i \in V$ as $d(\mathbf{x}_i) = \sum_j w(\mathbf{x}_i, \mathbf{x}_j)$. If $\mathbf{D}$ is the diagonal matrix with elements $d(\mathbf{x}_i)$, then the graph Laplacian is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{2}$$

It is sometimes beneficial to use normalized versions of the graph Laplacian, such as a symmetric graph Laplacian [103].

For some data, it is more desirable to compute the weights directly by calculating pairwise distances. In this case, the efficiency can be increased by using parallel computing or by reducing the dimension of data. Then, a graph is often made sparse using, e.g., thresholding or a nearest neighbors technique, resulting in graph where most of the edge weights are zero. Thus, the number of needed computations is reduced. Overall, a nearest neighbor graph can be computed efficiently using the kd-tree code of VLFeat library [3]. In particular, for the nearest neighbor technique, vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected only if $\mathbf{x}_i$ is among the N_n nearest neighbors of $\mathbf{x}_j$ or if $\mathbf{x}_j$ is among the N_n nearest neighbors of $\mathbf{x}_i$. Otherwise, $w(\mathbf{x}_i, \mathbf{x}_j)$ is set to 0.

For very large data sets, one can efficiently construct an approximation to the full graph using e.g. sampling-based approaches, such as the fast Nyström Extension method [33].

2.3 Semi-supervised setting

Despite the tremendous accomplishments of machine learning, its success depends on a sufficient amount of labeled samples. However, obtaining enough labeled data is difficult as it is time-consuming and expensive. Therefore, labeled data is scarce for most applications.

However, unlabeled data is usually easier and less costly to obtain than labeled data. Therefore, it is advantageous to use a semi-supervised setting, which uses both labeled and unlabeled data to construct the graph in order to reduce the amount of labeled data needed for good accuracy. In fact, the use of unlabeled data for graph construction allows one to obtain structural information of the data. Overall, for most graph-based semi-supervised methods, the majority of data embedded into a graph is unlabeled data. This paper derives methods which use a semi-supervised setting of this kind.

3 Methods

3.1 Background and related graph laplacian methods

3.1.1 Manifold learning

For the derivation of the MML method, let K be the number of classes, $\mathscr{L}$ be the set of labeled vertices, and $\mathscr{U}$ be the set of unlabeled vertices. We assume that $\mathscr{L}$ is drawn from the joining distribution P on $V \times \mathbb{R}$, while $\mathscr{U}$ is drawn from the marginal distribution P_V of P. We also assume that the conditional distribution $P(y|\mathbf{x})$ varies smoothly in the intrinsic geometry generated by P_V, where $y \in [1, K]$ and $\mathbf{x} \in V$.

In graph-based methods, information about labeled data and the geometric structure of the marginal distribution P_V of the unlabeled samples is incorporated into the problem:

$$f^* = \arg\min_{f \in \mathscr{H}_M} \frac{1}{|\mathscr{L}|} \sum_{\mathbf{x}_i \in \mathscr{L}} J(f, \mathbf{x}_i, y_i) + \gamma_A \|f\|_M^2 + \gamma_I \|f\|_I^2, \tag{3}$$

where the Mercer kernel $M : V \times V \to \mathbb{R}$ uniquely defines a reproducing kernel Hilbert space (RKHS) $\mathscr{H}_M$ with the corresponding norm $\|.\|_M$, J is a loss function which gives rise to different types of regularization problems, $\gamma_A > 0$, $\gamma_I > 0$, and $\|f\|_I^2$ is an additional regularizer that reflects the intrinsic geometry of P_V. The solution f^* to (3) can be described using the classical representer theorem [1]:

$$f^*(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathscr{L}} \alpha_i M(\mathbf{x}_i, \mathbf{x}) + \int_{\mathscr{S}} \alpha(\mathbf{z}) M(\mathbf{x}, \mathbf{z}) dP_V(\mathbf{z}), \tag{4}$$

where $\mathscr{S}$ is the support of the marginal distribution P_V [9].

In practice, that marginal distribution is unknown. In spite of that, one could empirically estimate $\|f\|_I$ by making use of the weighted graph as discussed in Section 2.2. With the pre-defined graph Laplacian matrix $\mathbf{L}$, the manifold regularizer $\|f\|_I^2$ can be empirically estimated [9] as

$$\|f\|_I^2 = \sum_{i,j=1}^{n} \big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big)^2 w_{ij} = \mathbf{f}^T \mathbf{L} \mathbf{f}, \tag{5}$$

where $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$.

The ambient norm $\|.\|_M$ and the intrinsic norm $\|.\|_I$ in (3) can be integrated in one term under the warped kernel $\tilde{M}$ [99]. This kernel defines an alternative reproducing kernel Hilbert space $\mathscr{\tilde{H}}$ by considering a modified inner product:

$$\langle f, g \rangle_{\mathscr{\tilde{H}}_{\tilde{M}}} = \langle f, g \rangle_{\mathscr{H}_M} + \mathbf{f}^T \mathbf{P} \mathbf{g}, \tag{6}$$

where $\mathbf{P}$ is a positive semi-definite matrix defined on labeled and unlabeled data, $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$ and $\mathbf{g} = [g(\mathbf{x}_1), g(\mathbf{x}_2), \cdots, g(\mathbf{x}_n)]^T$. With $\langle ., . \rangle_{\mathscr{\tilde{H}}_{\tilde{M}}}$, the warped kernel $\tilde{M}$ is shown in [99] to have the following representation:

$$\tilde{M}(\mathbf{x}, \mathbf{z}) = M(\mathbf{x}, \mathbf{z}) - \mathbf{M}_{\mathbf{x}}^T (\mathbf{I} + \mathbf{PM})^{-1} \mathbf{P} \mathbf{M}_{\mathbf{z}}, \tag{7}$$

where $\mathbf{M} = [m_{ij}]$ is the Gram matrix with $m_{ij} = M(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{M}_{\mathbf{x}}$ denotes the vector $(M(\mathbf{x}_1, \mathbf{x}), M(\mathbf{x}_2, \mathbf{x}), \cdots, M(\mathbf{x}_n, \mathbf{x}))^T$, and $\mathbf{M}_{\mathbf{z}}$ denotes the vector $(M(\mathbf{z}_1, \mathbf{x}), M(\mathbf{z}_2, \mathbf{x}), \cdots, M(\mathbf{z}_n, \mathbf{x}))^T$.

The regularization problem for the warped kernel $\tilde{M}$ is:

$$f^* = \arg\min_{f \in \mathscr{\tilde{H}}_{\tilde{M}}} \frac{1}{|\mathscr{L}|} \sum_{\mathbf{x}_i \in \mathscr{L}} J(f, \mathbf{x}_i, y_i) + \gamma_A \|f\|_{\tilde{M}}^2. \tag{8}$$

Problem (8) exploits the intrinsic geometry of P_V via the data-dependent kernel $\tilde{M}$ but still makes use of the classical regularization solvers. In fact, the classical representer theorem [1] allows f^* in (8) to be expressed as:

$$f^*(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathscr{L}} \alpha_i \tilde{M}(\mathbf{x}, \mathbf{x}_i). \tag{9}$$

In practice, $\{\alpha_i\}$ are numerically determined by an appropriate optimization solver, e.g., [28].

3.1.2 MBO reduction

For the derivation of the MMBO method, we first note that a typical learning algorithm involves finding an optimal label matrix $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_N)^T$ associated with data elements, where $\mathbf{u}_i \in \mathbb{R}^K$ represents the probability distribution over the classes for data element $\mathbf{x}_i$; the i^{th} row of $\mathbf{U}$ is set to $\mathbf{u}_i$. The vector $\mathbf{u}_i$ is an element of the Gibbs simplex:

$$\Sigma^K := \{(z_1, \ldots, z_K) \in [0, 1]^K \,|\, \sum_{k=1}^{K} z_k = 1\}, \tag{10}$$

where K is the number of classes. Moreover, the k^{th} vertex of the simplex is given by the unit vector $\mathbf{e}_k$.

A general form of a graph-based problem for data classification is the minimization of $E(\mathbf{U}) = R(\mathbf{U}) + \text{Fid}(\mathbf{U})$, where $\mathbf{U}$ is the data classification function, $R(\mathbf{U})$ is a graph-based regularization term incorporating the graph weights, and $\text{Fid}(\mathbf{U})$ is a term incorporating labeled points.

Not surprisingly, the choice of the regularization term has non-trivial consequences in the final accuracy. In [37], Garcia et al. successfully take for the regularization term a multiclass graph- based Ginzburg-Landau (GL) functional:

$$\text{GL}(\mathbf{U}) = \frac{\epsilon}{2}\langle \mathbf{U}, \mathbf{L}\mathbf{U}\rangle + \frac{1}{2\epsilon}\sum_i \left(\prod_{k=1}^{K}\frac{1}{4}\|\mathbf{u}_i - \boldsymbol{e}_k\|_{L_1}^2\right). \quad (11)$$

Here, $\epsilon > 0$, $\mathbf{L}$ is a normalized graph Laplacian, K is the number of classes, $\langle \mathbf{U}, \mathbf{L}\mathbf{U}\rangle = \text{trace}(\mathbf{U}^T\mathbf{L}\mathbf{U})$, $\mathbf{u}_i$ is the i^{th} row of $\mathbf{U}$, $\boldsymbol{e}_k$ is an indicator vector of size K with one in the k^{th} component and zero elsewhere. The first (smoothing) term in (11) measures variations in the vector field, while the second (potential) term in (11) drives the system closer to the vertices of the simplex.

While it is possible to develop a convex splitting scheme to minimize the graph-based multiclass GL energy [37], a more efficient technique involves MBO reduction. Specifically, if one considers the minimization of the GL functional plus a fidelity term (consisting of a fit to elements of known class) in the continuous case, one can apply L_2 gradient descent resulting in a modified Allen-Cahn equation. If a time-splitting scheme is then applied, one obtains a procedure where one alternates between propagation using the heat equation with a forcing term and thresholding. In such a state, the resulting procedure has similar elements to the MBO scheme [72], which evolves an interface by mean curvature, in a continuous, rather than graph-based, setting. The procedure can then be transferred to a graph-based setting using [37, 68, 69]. Moreover, in order for the scheme to be applicable to the multiclass case, one can convert the thresholding operation to the displacement of the vector field variable towards the closest vertex in (10) [37, 68, 69].

3.2 The derivation of the multiscale setting and the proposed methods

3.2.1 Multiscale graph laplacian operator

The dominance of multiscale information over the single one has been proved in various biophysics-related works, such as those involving thermal fluctuation predictions [85, 111] and binding affinity predictions [76]. Therefore, it is promising to explore how the multiscale approach can improve the accuracy of graph-based data classification. We examine a novel multiscale graph Laplacian in the form of

$$\mathbf{L}_{\text{multiscale}} = \sum_{t=0}^{m} c_t \mathbf{L}_t^{p_t}, \quad (12)$$

where $p_t > 0$, $c_t > 0$, and L_t is an extended Laplacian matrix defined by $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$, where $\mathbf{D}_t$ is a degree matrix, and $\mathbf{W}_t$ is an extended adjacent graph edge matrix

$$[\mathbf{W}_t]_{ij} = \frac{1}{\sqrt{\sigma_t}} H_t\left(\frac{||x_i - x_j||}{\sigma_t}\right) e^{-\frac{||x_i - x_j||^2}{\sigma_t^2}}, \quad (13)$$

where $\sigma_t > 0$ and H_t is the t^{th} order Hermite polynomial. Usually, *only two or three multiscale Laplacian terms* in (12), i.e., $m = 1$ or $m = 2$, are needed to obtain a significant improvement in accuracy; by setting $m = 0$ and $c_0 = 1$, one can restore the regular graph Laplacian discussed in (2). In this formulation, σ_t is automated scale filtration parameter that controls the shape of a submanifold for a data set, while c_t weighs contributions from different scales. The parameters c_t and σ_t may vary for different Hermite polynomials.

In case of large data for which computing all the graph weights can be computationally expensive, one can use the Nyström extension method [10, 33, 34] to compute approximations to the few smallest eigenvalues and corresponding eigenvectors of the multiscale graph Laplacian while calculating only a small fraction of the graph weights. We will modify the Nyström procedure to incorporate the new multiscale graph Laplacian $\mathbf{L}_{\text{multiscale}}$. In this case, the weights in the procedure are computed using

$$\mathbf{W}_{\text{multiscale}} = \sum_{t=0}^{m} c_t \mathbf{W}_t^{p_t}, \quad (14)$$

where, in most cases, $m = 1$ or $m = 2$ is enough to obtain a significant accuracy improvement.

When the number of data elements is not too large, one can compute the eigenvectors via the Rayleigh-Chebyshev method [6] or the Shifted Block Lanczos algorithm [42].

3.2.2 Multikernel manifold learning (MML) scheme

In multikernel manifold learning (MML), the multiscale Laplacian matrices proposed in (12) is employed to form N_n-nearest neighbors subgraphs. By setting $\mathbf{P} = \frac{\gamma_I}{\gamma_A}\mathbf{L}_{\text{multiscale}}$ in (7), we attain an MML scheme enabling the reconstruction of the regularization problem presented in (3). Even with the integration of multiscale Laplacian operator into the data kernel, the manifold learning algorithms still retains its classical representation as presented in (8). One, therefore, could utilize traditional solvers to derive the

multiscale manifold learning's optimizer [99]. The MML procedure is summarized as Algorithm 1.

Require: labeled data $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_i$, where y_i is the label of $\mathbf{x}_i$, unlabeled data $\mathscr{U} = \{\mathbf{x}_j\}_j$, N_n (# of nearest neighbors), $m+1$ (# of scales), where 2 or 3 scales is usually sufficient, $\{c_t\}_{t=0}^m$ (Laplacian matrix coefficients), $\{p_t\}_{t=0}^m$ (matrix powers), $\{\sigma_t\}_{t=0}^m$ (kernel scales), and $\gamma_I > 0$, $\gamma_A > 0$ (scalars).

Ensure: Estimated optimizer f^*, where $f^*(\mathbf{x})$ is the prediction for $\mathbf{x}$.

1: Construct $m+1$ multiscale subgraphs with N_n nearest neighbors with weights $[\mathbf{W}_t]_{ij}$ for $t = 0, ..., m$, where it is usually sufficient to use $m = 1$ or $m = 2$, i.e. two or three scales.

2: Select the kernel $M(\mathbf{x}, \mathbf{x}_i)$, e.g., radial basis function kernel or a Gaussian kernel.

3: Compute the Gram matrix $\mathbf{M} = [m_{ij}]$ with $m_{ij} = M(\mathbf{x}_i, \mathbf{x}_j)$.

4: Compute the multiscale Laplacian $\mathbf{L}_{\text{multiscale}}$ using (12) and $\{c_t\}_{t=1}^m$, $\{p_t\}_{t=0}^m$ and $\{\sigma_t\}_{t=0}^m$.

5: Formulate the warped kernel $\tilde{M}(\mathbf{x}, \mathbf{x}_i)$ using (7) and $\mathbf{P} = \mathbf{L}_{\text{multiscale}}$.

6: Solve for optimizer of (8) using an SVM quadratic programing solver for soft margin loss, e.g., [28].

Algorithm 1 MML Algorithm (multiscale).

3.2.3 Multiscale MBO (MMBO) scheme

Our proposed MMBO scheme uses a semi-implicit approach where the multiscale Laplacian term is computed implicitly due to the stiffness of the operator which is caused by a wide range of its eigenvalues. An implicit term here is needed since an explicit scheme requires all scales of the eigenvalues to be resolved numerically.

To derive the MMBO scheme, let $\mathbf{U}$ represent a matrix where each row is a probability distribution of each data element over the classes and let $\mathbf{u}_i$ represent the i^{th} row of $\mathbf{U}$. In addition, let N be the number of data set elements, K be the number of classes, dt>0, and $\boldsymbol{\mu}$ be a vector which takes a value μ in the i^{th} place if $\mathbf{x}_i$ is a labeled element and 0 otherwise. Moreover, let $\mathbf{U}_{\text{labeled}}$ be the following matrix: for rows corresponding to labeled points, the entry corresponding to the class of the labeled point is set to 1. All other entries of the matrix are set to 0. Lastly, let $\boldsymbol{\mu}\cdot(\mathbf{U}-\mathbf{U}_{\text{labeled}})$ indicate row-wise multiplication by a scalar.

As described in Section 3.1.2, if one considers the minimization of a GL functional plus a fit to elements of known class in the continuous case, an L_2 gradient descent results in a modified Allen-Cahn equation. If a time-splitting scheme is then applied, one obtains a procedure where one alternates between propagation using the heat equation with a forcing term and thresholding. The scheme can then be transferred to a graph-based setting and the Laplace operator can be replaced by a graph-based multiscale Laplacian. The thresholding can be changed to the displacement of the variable towards the closest vertex in (10). A projection to the simplex is then necessary before the displacement step.

Our proposed MMBO procedure thus consists of the following procedure. Starting with an initial guess for $\mathbf{U}$, obtain the next iterate of $\mathbf{U}$ via the following three steps:

1. Multiscale heat equation with a forcing term:
$$\mathbf{U}^{n+\frac{1}{2}} = \mathbf{U}^n - \text{dt}\{\mathbf{L}_{\text{multiscale}}\mathbf{U}^{n+\frac{1}{2}} + \boldsymbol{\mu}\cdot(\mathbf{U}^n - \mathbf{U}_{\text{labeled}})\},$$
where $\boldsymbol{\mu}$ is a vector which takes a value μ in the i^{th} place if $\mathbf{x}_i$ is a labeled element and 0 otherwise, and the term $\boldsymbol{\mu}\cdot(\mathbf{U}^n-\mathbf{U}_{\text{labeled}})$ indicates row-wise multiplication by a scalar.
2. Projection to simplex: Each row of $\mathbf{U}^{n+\frac{1}{2}}$ is projected onto the simplex using [26].
3. Displacement: $\mathbf{u}_i^{n+1} = \boldsymbol{e}_k$, where $\mathbf{u}_i^{n+1}$ is the i^{th} row of $\mathbf{U}^{n+1}$, and $\boldsymbol{e}_k$ is the indicator vector (with a value of 1 in the k^{th} place and 0 elsewhere) associated with the vertex in the simplex closest to the i^{th} row of the projected $\mathbf{U}^{n+\frac{1}{2}}$ from step 2.

This implicit scheme allows the evolution of $\mathbf{U}$ to be numerically stable regardless of the time step dt, in spite of the "stiffness" of the differential equations which could otherwise force dt to be impractically small.

One can compute $\mathbf{U}^{n+\frac{1}{2}}$ very efficiently using spectral techniques and projections onto a low-dimensional subspace spanned by a small number of eigenfunctions in the following manner, where $\mathbf{I}$ is the identity:

$$\mathbf{U}^{n+\frac{1}{2}} = \mathbf{X}_{\text{multiscale}}\,(\mathbf{I} + \text{dt}\,\boldsymbol{\Lambda}_{\text{multiscale}})^{-1}\,\mathbf{X}_{\text{multiscale}}^T\mathbf{U}_{\text{update}}, \tag{15}$$

where $\mathbf{U}_{\text{update}} = \mathbf{U}^n - \text{dt}\,\boldsymbol{\mu}\cdot(\mathbf{U}^n - \mathbf{U}_{\text{labeled}})$, $\mathbf{X}_{\text{multiscale}}$ is an $N \times N_e$ truncated matrix retaining only $N_e << N$ smallest eigenvectors of the multiscale graph Laplacian $\mathbf{L}_{\text{multiscale}}$, and $\boldsymbol{\Lambda}_{\text{multiscale}}$ is a $N_e \times N_e$ truncated diagonal matrix retaining only the smallest eigenvalues of the multiscale Laplacian $\mathbf{L}_{\text{multiscale}}$ along the diagonal.

The proposed MMBO procedure is detailed as Algorithm 2. It is important to note that in the MMBO method, the graph weights are only used to compute the few eigenvectors and eigenvalues of the multiscale graph Laplacian, and the multiscale MMBO procedure themselves do not involve graph weights. This crucial property allows one to use the Nyström extension procedure [10, 33, 34] to approximate the extremal eigenvectors of the Laplacian by only computing a small portion of the graph weights; this enables one to apply the multiscale models very efficiently on large data.

For initialization, the rows of $\mathbf{U}$ corresponding to labeled points are set to the vertices of the simplex corresponding to the known labels, while the rows of $\mathbf{U}$ corresponding to the rest of the points initially represent random probability distributions over the classes.

The energy minimization proceeds until a steady state condition is reached. One way of determining a steady state condition is to stop the calculation when, for a positive constant $\eta > 0$,

$$\frac{\max_i \|\mathbf{u}_i^{n+1} - \mathbf{u}_i^n\|^2}{\max_i \|\mathbf{u}_i^{n+1}\|^2} < \eta. \tag{16}$$

The final classes are obtained by assigning class k to node i if $\mathbf{u}_i$ is closest to vertex $\boldsymbol{e}_k$ on the Gibbs simplex.

3.3 Computation of eigenvalues and eigenvectors of the multiscale graph laplacian

The MMBO method requires one to compute a few of the smallest eigenvalues and the corresponding eigenvectors of the multiscale graph Laplacian to form $\mathbf{X}_{\text{multiscale}}$. We examine and use three techniques for this task. Nyström extension [10, 33, 34] is the preferred method for very large data.

3.3.1 Nyström extension for fully connected graphs

Nyström extension [10, 33, 34] is a matrix completion method, and it performs faster than many other techniques because it computes approximations to eigenvectors and eigenvalues using *much smaller* submatrices of the original matrix.

Note that if λ is an eigenvalue of $\hat{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, then $1 - \lambda$ is an eigenvalue of the symmetric Laplacian $\mathbf{L_s} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, and the two matrices have the same eigenvectors. Thus, one can use Nyström extension to calculate approximations to the eigenvectors of $\hat{\mathbf{W}}$ and thus of $\mathbf{L_s}$.

Now, consider the problem of approximating the extremal N_e eigenvalues and eigenvectors of a full graph $\hat{\mathbf{W}}$ and let $\hat{w}(x_i, x_j) = \hat{\mathbf{W}}_{ij}$. Nyström extension [10, 33, 34] approximates the eigenvalue equation using a quadrature rule and $N_e << N$ randomly chosen interpolation points from V, which represents data elements. Denote the set of N_e randomly chosen points by $X = \{x_i\}_{i=1}^{N_e}$ and its complement by Y. Partitioning V into $V = X \cup Y$ and letting $\phi_k(x)$ be the the k^{th} eigenvector of W and λ_k be its associated eigenvalue, we obtain:

$$\sum_{x_j \in X} \hat{w}(y_i, x_j)\phi_k(x_j) = \lambda_k \phi_k(y_i) \forall y_i \in Y, \forall k \in 1, ..., N_e. \tag{17}$$

Require: labeled data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_i$, where y_i is the label of $\mathbf{x}_i$, unlabeled data $\mathcal{U} = \{\mathbf{x}_j\}_j$, N_n (# of nearest neighbors), $m + 1$ (# of scales), $\{c_t\}_{t=0}^m$ (Laplacian matrix coefficients), $\{p_t\}_{t=0}^m$ (matrix powers), $\{\sigma_t\}_{t=0}^m$ (kernel scales), dt > 0, N (# of data set elements), $N_e << N$ (# of eigenvectors to be computed), N_t (maximum # of iterations), μ (an $N \times 1$ vector which takes a value μ in the i^{th} place if x_i is a labeled element and 0 otherwise).

Ensure: out $= \mathbf{U}^{\mathbf{end}}$; the i^{th} row of $\mathbf{U}^{\mathbf{end}}$ is a probability distribution of data element $\mathbf{x}_i$ over the classes.

1. For larger data, go to Step 4. For smaller data, go to Step 2.

2: Construct $m+1$ multiscale subgraphs with N_n nearest neighbors with weights $[\mathbf{W}_t]_{ij}$ for $t = 0, ..., m$, where it is usually sufficient to use $m = 1$ or $m = 2$, i.e. two or three scales.

3: Compute the multiscale Laplacian $\mathbf{L}_{\text{multiscale}}$ using (12) and $\{c_t\}_{t=0}^m$, $\{p_t\}_{t=0}^m$ and $\{\sigma_t\}_{t=0}^m$.

4: Compute $\mathbf{U}_{\text{labeled}}$, $\mathbf{\Lambda}_{\text{multiscale}}$ and $\mathbf{X}_{\text{multiscale}}$ as described in Section 3.2.3 and using $N_e << N$. For smaller data, use methods such as [6]. For larger data, use Nyström extension [10,33,34].

5: Complete the following steps: starting with $n = 1$.

for $i = 1 \rightarrow N$ **do**

$\mathbf{U}_{ik}^0 \leftarrow rand((0, 1))$, $\mathbf{u}_i^0 \leftarrow projectToSimplex(\mathbf{u}_i^0)$ using [26], where i^{th} row of $\mathbf{U}^0$.

If $\mu_i > 0$, $\mathbf{U}_{ik}^0 \leftarrow \mathbf{U}_{\text{labeled}ik}$

end for

$\mathbf{B} \leftarrow (\mathbf{I} + \text{dt}\Lambda_{\text{multiscale}})^{-1}\, \mathbf{X}_{\text{multiscale}}{}^T$

while Stop criterion not satisfied$or n > N_t$ **do**

$\mathbf{C} \leftarrow \mathbf{U}^n - \text{dt}\, \boldsymbol{\mu} \cdot (\mathbf{U}^n - \mathbf{U}_{\text{labeled}})$

$\mathbf{A} \leftarrow \mathbf{BC}$

$\mathbf{U}^{n+1} \leftarrow \mathbf{X}_{\text{multiscale}}\mathbf{A}$

for $i = 1 \rightarrow N$ **do**

$\mathbf{u}_i^{n+1} \leftarrow projectToSimplex(\mathbf{u}_i^{n+1})$ using [26]

$\mathbf{u}_i^{n+1} \leftarrow \boldsymbol{e}_k$, where k is closest simplex vertex to $\mathbf{u}_i^{n+1}$

end for

The matrix $\mathbf{U}^{n+1}$ is such that its i^{th} row is $\mathbf{u}_i^{n+1}$.

$n \leftarrow n + 1$

end while

Algorithm 2 MMBO Algorithm (multiscale).

This system cannot be solved directly since the eigenvectors are unknown; thus, the N_e eigenvectors of $\hat{\mathbf{W}}$ are approximated using much smaller submatrices of $\hat{\mathbf{W}}$.

The efficiency of Nyström extension lies with the following fact: when computing the N_e eigenvalues and

eigenvectors of an $N \times N$ matrix, where N is large, the algorithm approximates them using much smaller matrices, the largest of which has dimension $N \times N_e$, where $N_e << N$. In particular, when the method is applied to $\mathbf{W}$ or $\hat{\mathbf{W}}$, *only a small portion of the weight matrix* $\mathbf{W}$ *or* $\hat{\mathbf{W}}$ *needs to be computed*. In our experience, $N_e = 100$ or $N_e = 200$ were good choices.

If the number of scales is $m + 1$, the complexity of the Nyström procedure is $O(NN_e(m + 1))$, which is linear in N.

3.3.2 Rayleigh-Chebyshev method

The Rayleigh-Chebyshev method [6] is a fast algorithm for finding a small subset of eigenvalues and eigenvectors of sparse symmetric matrices, such as a symmetric graph Laplacian which can be made sparse using techniques such as N_n-nearest neighbors. The method is a modification of an inverse subspace iteration procedure and uses adaptively determined Chebyshev polynomials.

3.3.3 A shifted block Lanczos algorithm

A shifted block Lanczos algorithm [42], as well as other variations of the Lanczos method [86] that is an adaptation of power methods, are efficient techniques for solving sparse symmetric eigenproblems and for finding a few of the extremal eigenvalues. They can be used to find a subset of the eigenvalues and eigenvectors of the symmetric graph Laplacian which can be made sparse using N_n-nearest neighbors.

4 Results and discussion

4.1 Data sets

In this work, we validate the proposed MML and MMBO methods against three common data sets:

- G50C is an artificial data set inspired by [41] and generated from two normal covariance Gaussian distributions. This data set has 550 data points located in $\mathbb{R}^{50}$ and two labels $\{-1, +1\}$.
- USPST data set includes images of handwritten digits taken from the USPS test data set. This data has 2007 images to be classified into ten labels corresponding to ten numbers from 0 to 9.
- Mac-Win data set categorizes documents, taken from 20-Newsgroups data, into 2 classes: mac or windows [101]. This set has 1946 elements and each element is represented by a vector in $\mathbb{R}^{7511}$.
- WebKB data set is taken from the web documents of the CS department of four universities and has been used extensively. It has 1051 data samples and two labels: course and non-course. There are two ways to describe each web document: the textual content of the webpage (called page representation), and the anchor text on hyperlinks pointing from other webpages to the current one. The data points with page representation are in $\mathbb{R}^{3000}$, while the ones with link representation belong to $\mathbb{R}^{1840}$. When we combine two different kinds of representations, we achieve the data points in $\mathbb{R}^{4840}$.
- α, β-protein data set consists of three different protein domains, namely alpha proteins, beta proteins, and mixed alpha and beta proteins, classified based on protein secondary structures [17]. This data has 900 biomolecules, and each family has 300 instances.

The details of the data sets are outlined in Table 1.

4.2 Hyperparameters selection

In the MMBO setting, for each data point, we do not compute the complete graph but instead construct a N_n-nearest neighbor graph for the calculation efficiency. The parameter l is one of the hyperparameters and is selected on a case by case basis. Moreover, as discussed in Section 2.2, the weight function used is the Gaussian kernel $w(\mathbf{x}_i, \mathbf{x}_j) = \exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/\sigma^2)$. Here, the scalar σ is optimized so that it perfectly fits the labeled set information. In the multiscale approach, each kernel is assigned different σ values depending on the outcome of hyperparameter selection. Overall, due to the random initialization of the non-labeled points, we use the same random seed for all the experiments in this work for reproducible purposes.

The Nyström extension method [10, 33, 34] allows for fast computations even in case of larger data since this approach approximates the eigenvalues and eigenvectors of the origin matrix using much smaller matrices randomly selected from the bigger ones. Thus, only a small portion of the graph weights need to be computed. However, in case of smaller data, it is often more advantageous to use methods such as [6] which can directly compute the eigenvalues and eigenvectors. Therefore, to obtain optimal results, we employ the Rayleigh-Chebyshev procedure [6] (see Section 3.3.2) for our experiments. This method is well-known for efficiently calculating the smallest eigenvectors of a sparse symmetric matrix. The hyperparameters of the MMBO models are the number of leading eigenvalues (N_e), the time step for solving heat equation (dt), the constraint constant on fidelity term ($\boldsymbol{\mu}$), and the number of iterations (N_t).

Table 1 Data sets used in the experiments

Data set	No. of classes	Sample dim.	No. of data elements	No. of labeled data
G50C	2	50	550	50
USPST	10	256	2007	50
Mac-Win	2	7511	1946	50
WebKB (page)	2	3000	1051	12
WebKB (link)	2	1840	1051	12
WebKB (page+link)	2	4840	1051	12
α, β-protein	3	50	900	720

The hyperparameter selection for MML model is carried out in a similar fashion as that of the MMBO algorithm. The tunning parameters are: the number of nearest neighbors (N_n), the scaler factor (σ), the penalty coefficient (γ_A), the manifold regularizer constraint (γ_I), and the Laplacian degree (p). The optimizer is solved using the primal SVM solver [64]. The optimal hyperparameters of the proposed methods are documented in the Supporting Information.

4.3 Performance and discussion

4.3.1 Non-biological data sets

The non-biological data sets we used for our experiments are the G50C, USPST, Mac-Win, and WebKB data sets. In the experiments involving these data sets, we utilize the original representations without carrying out any feature generation procedures. In addition, following the previous work [23, 99], we only consider accuracy as the main evaluation metric for the G50C, USPST, and Mac-Win data sets. For WebKB, we compute the Precision/Recall Breakeven Point (PRBEP) due to its imbalanced labeling, but also use the classification accuracy to compare to more recent methods.

The results for non-biological data sets are shown in Figs. 1 and 2. In most experiments with non-biological data, the proposed MMBO method is clearly the most dominant. The other proposed model, the MML method, is the second best model with promising performances.

In all cases, the results of the proposed MML and MMBO methods show promising improvements from non-multiscale frameworks. Specifically, the best performances of the algorithms are achieved with three kernels. In particular, there is a significant accuracy improvement from single kernel to two kernel architectures on the USPST data (from 86.11% to 90.57% for the MML model, and from 86.55% to 88.65% for the MMBO model) and Mac-Win data (from 89.98% to 90.01% for the MML model, and from 92.06% to 93.49% for the MMBO model). The improvement from single kernel to multi-kernel learning is less for the G50C and WebKB data, but that is to be expected since G50C is a small data set consisting of 550 samples. Furthermore, it is an artificial data set drawn from two unit covariance normal distributions. As a result, a single kernel is enough to capture the crucial structure of data. Moreover, the WebKB data poses a challenge for multiscale learning due to its imbalanced data.

In almost all experiments conducted for this paper, the proposed MMBO and MML models obtain the best results, mostly with three kernels. In particular, for the G50C data, the MMBO method achieves the best accuracy (95.06%), but the MML method is still comparable with its accuracy being 94.56%. Moreover, the superior performance of our proposed algorithms over the state-of-the-art models is also displayed in the case of the more complex USPST data, a set of handwritten digit images with 1440 samples. While the proposed MML algorithm obtains the best accuracy at 90.57%, the MMBO method with three-kernel information still obtains a good accuracy of 88.73%. The other published approaches, such as LapRLS [99], obtain lower accuracies. For Mac-Win, the accuracies of our multi-scale models are slightly lower than those of ∇TSVM (94.3%) [23] and LDS (94.9%) [23]. The fact that there are only 1966 samples but the dimension of each sample is very high, i.e., 7511, might indicate noisy information which can reduce the performances of graph-based kernel models. For WebKB, our proposed methods perform extremely well. WebKB is the last data set in this category and has three different feature representations, namely, link, page and page+link. The overall performance of our proposed models is very encouraging. We see that using only one kernel already produces great results, with a little improvement in using multiple kernels. With the PRBEP metric, the best model is the MMBO method with 3 kernels which obtains a PRBEP at 96.22%, 97.93%, and 98.87% for the link, page, and page+link experiments, respectively. The MML method obtains the next best result with a PRBEP of 95.75%, 95.81%, and 95.84% for link, page, and page+link experiments, respectively. After the proposed methods, the next best result using the PRBEP metric is obtained with LapSVM [99]: 94.3%, 93.4%, and 94.9%. When using the classification accuracy and 105 labeled elements, the

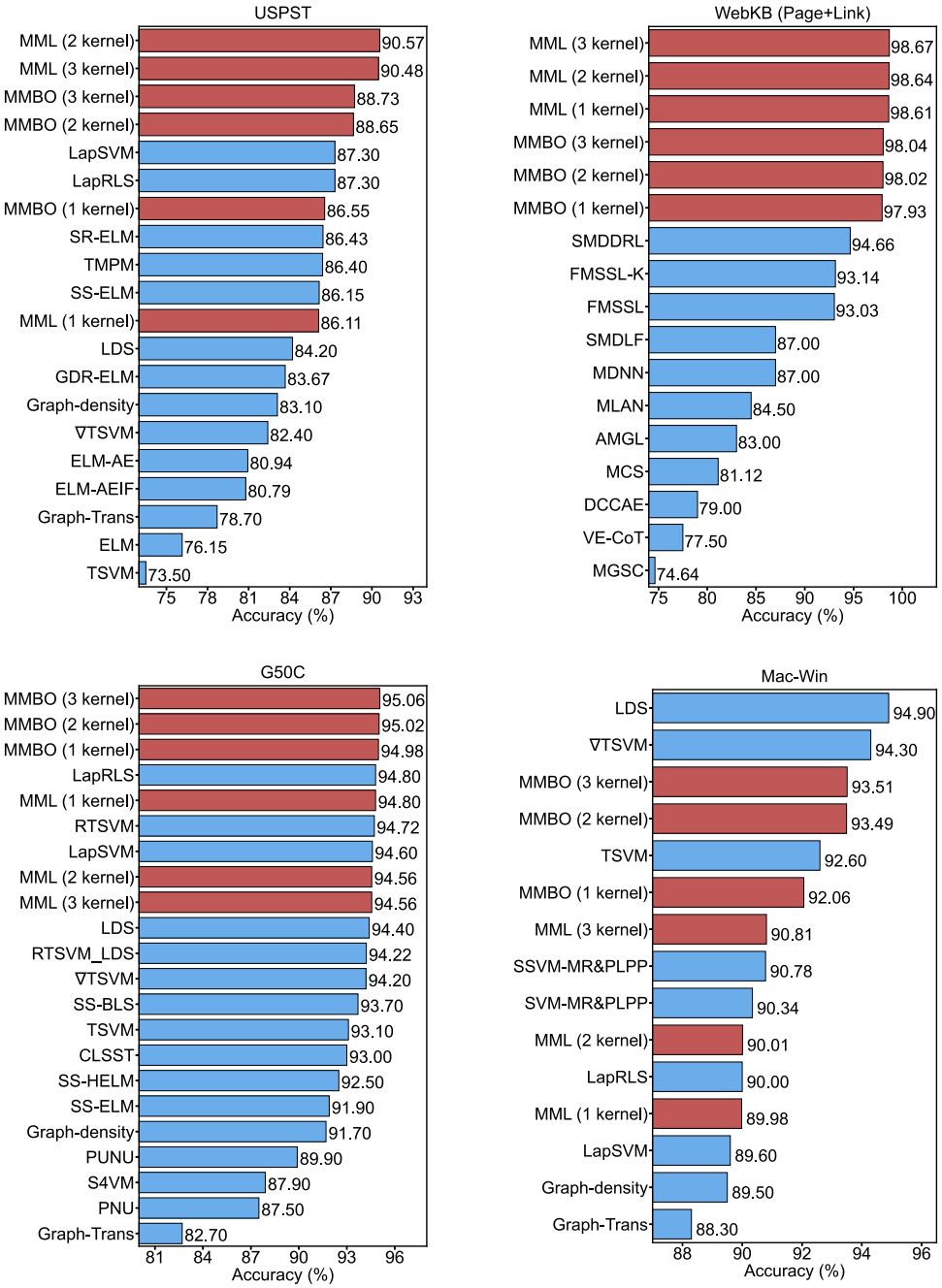

Fig. 1 Accuracy comparison of MML and MMBO with other methods on USPST, WebKB (Page+Link), G50C, and Mac-Win datasets. The proposed methods are in red, and other methods are in blue. We note that some of the comparison methods for USPST use more labeled samples than the proposed methods. Please refer to Section 4.4 for more details. See Fig. 2 for more comparison

MML model performs the best, with accuracies in the 98^{th} percentile.

4.3.2 Alpha and beta protein classification

We also tested the proposed multiscale learning models using biological data, such as data involving protein classification. In this data, based on the secondary structure, proteins are typically grouped into three classes, namely alpha helices, beta sheets, and mixed alpha and beta domains. Figure 3 plots the secondary-structure representations of 3 types of protein structures. The data, which consists of 900 structures equally distributed into three classes, was collected by Cang et al. [17] and taken from SCOPe (Structural Classification of Proteins-extended), an online database [35].

Five-fold cross validation is conducted to examine the performance of the proposed models. To preserve the unbiased information, in each fold, the test set consisted of 180 instances with 60 samples from each group. Overall,

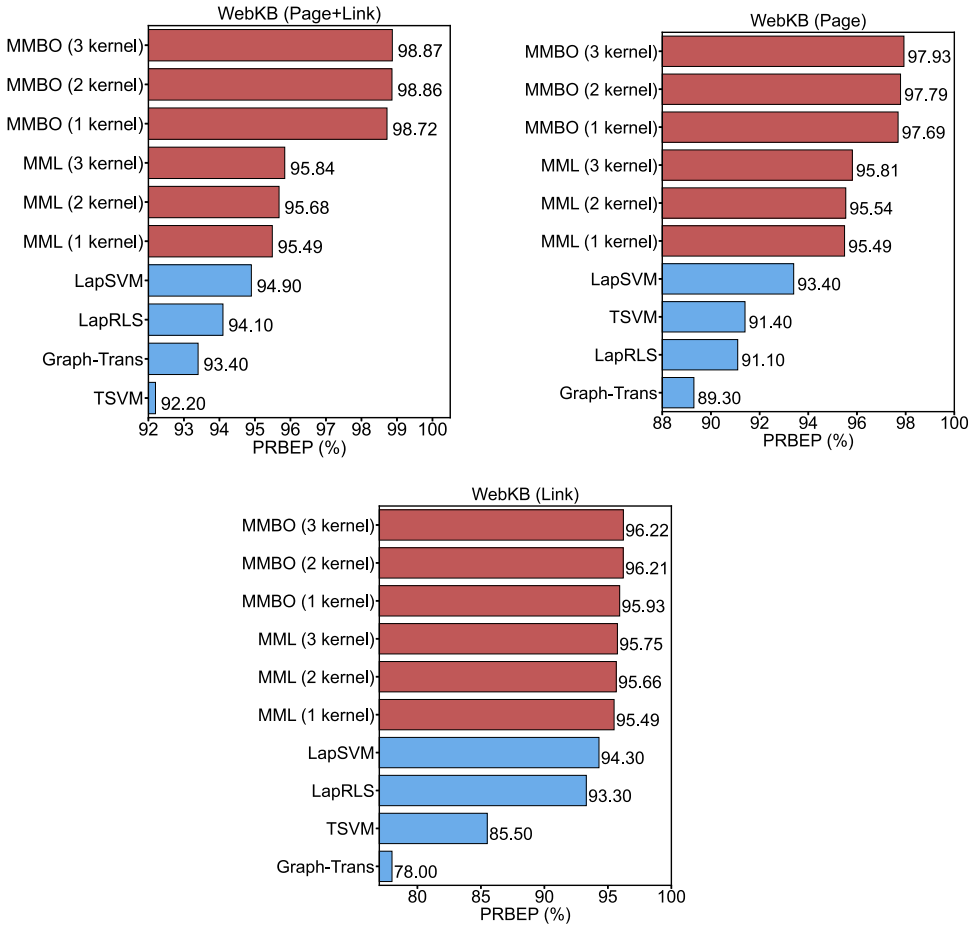

Fig. 2 PRBEP comparison of MML and MMBO with other methods on WebKB (Page+Link), WebKB (Page), and WebKB (Link) datasets. The proposed methods are in red, and other methods are in blue. See Fig. 1 for more comparison

the protein data sets originally provide the coordinates and atom types for each structure. However, feature generation is needed to translate such information to a vector format suitable for machine learning algorithms. Moreover, for this data, the feature generation has to sustain crucial physical and chemical interactions such as covalent and non-covalent bonds, electrostatic, hydrogen bonds, etc. In the past few years, we have developed numerous mathematical-based feature engineering models including geometric and algebraic graph [73, 77], differential geometry [75], persistent homology [16], and persistent graph [108] for representing 3D molecular information in low dimensional representations.

We employ our geometric graph representation in [77]. In order to represent the physical and chemical properties of a biomolecule, we consider four atom types, namely C_α, C, N, and O. In particular, the protein structures are described by vectors of 50 components. Overall, the details of the

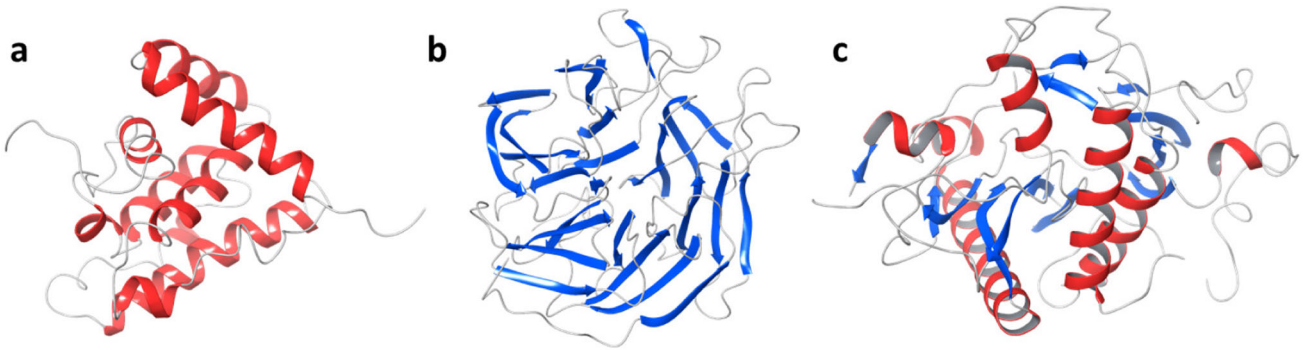

Fig. 3 Secondary-structure representations of proteins taken from α, β-protein data. Here, alpha helix is colored in red, beta sheet is colored in blue. a) Alpha protein (PDBID: 1WIX), b) Beta protein (PDBID: 3O4P), c) Mixed-alpha and beta protein (PDBID: 2CNQ). PDBID stands for protein data bank ID with experimental structures available at https://www.rcsb.org/

parameters for the feature generated approach is provided in the Supporting Information.

Both MMBO and MML models perform well. Moreover, similarly to previous experiments, multiscale information strengthens the accuracy of both MML and MMBO approaches. In fact, there is an encouraged improvement from the one kernel model to the two kernel model , i.e., 84% to 85% accuracy for the MMBO model. There is also an improvement in the MMBO method results using three kernels, i.e. 85.11%. For the MML method, there is a slight improvement by using multiple kernels. For this data, the MMBO method outperforms its counterpart, which indicates the versatility of the MMBO algorithm when dealing with a variety of data. All results are presented in Fig. 4.

4.4 Comparison algorithms

We compare our algorithms to many recent methods, most of which are from 2015 and later.

For WebKB data, we compare classification accuracy against recent methods such as semi-supervised multi-view deep discriminant representation learning (SMDDRL) [48], vertical ensemble co-training (VE-CoT) [54], auto-weighted multiple graph learning (AMGL) [82], multi-view learning with adaptive neighbors (MLAN) [79], deep canonically correlated autoencoder (DCCAE) [109], multi-view discriminative neural network (MDNN) [84], semi-supervised learning for multiple graphs by gradient flow (MGSC) [62], multi-domain classification w/ domain selection (MCS) [24], multi-view semi-supervised learning (FMSSL, FMSSL-K) [115], and semi-supervised multi-modal deep learning framework (SMDLF) [27]. Our results are obtained using 105 labels, and using the classification accuracy metric. Results for SDMDRL, VE-CoT, AMGL, MLAN, SMDLF, DCCAE and MDNN are from [48], the results for MGSC and MCS are from [62], and the results for FMSSL and FMSSL-K are from [115]. All methods use 105 labels.

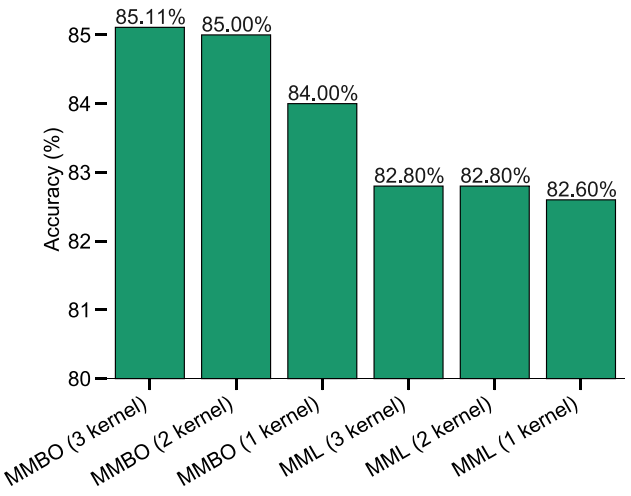

Fig. 4 The performances of MMBO and MML models on the protein classification data set

For USPST, we compare against recent methods such as transductive minimax probability machines (TMPM) [44], semi-supervised extreme learning machines (SS-ELM) [43], graph embedding-based dimension reduction with extreme learning machines (GDR-ELM) [112], extreme learning machine auto-encoder (ELM- AE) [53], extreme learning machine auto-encoder with invertible functions (ELM-AEIf) [113] and extreme learning machines for dimensionality reduction (SR-ELM) [5]. Our results are obtained using only 50 labels. The results for TMPM (with 50 labels) are from [44], the results for GDR-ELM, ELM-AE, ELM-AEIf and SR-ELM (with 150 labels) are from [112], and the result for SS-ELM (with 100 labels) are from [43].

For G50C, we compare against recent methods such as classtering (CLSST) [94], semi-supervised broad learning system (SS- BLS) [118], classification from positive and unlabeled data (PNU) [93], classification from unlabeled positive and negative data (PUNU) [93], semi-supervised extreme learning machines (SS-ELM) [43], semi-supervised hierarchical extreme learning machine (SS-HELM) [98], safe semi-supervised support vector machines (S4VM) [59], robust and fast transductive support vector machines (RTSVM, RTSVM- LDS) [21]. Our results are obtained using 50 labels. The result for CLSST is from [94], the results for SS-BLS, SS-ELM and SS-HELM are obtained from [118], the results for PNU, PUNU and S4VM are obtained from [93], and the results for RTSVM and RTSVM-LDS are obtained from [21]. All comparison methods use 50 labels.

For Mac-Win, we compare against recent methods such as support vector machines with manifold regularization and partially labeling privacy protection (SVM-MR&PLPP) [78] and a scalable version (SSVM-MR&PLPP) [78]. These results are obtained from [78]. All methods use 50 labels.

We also compare results for all data sets with slightly older methods such as transductive graph methods (Graph-Trans), closely related to [8, 119, 122], transductive support vector machines (TSVM) [50], support vector machines on a graph-distance derived kernel (Graph-density) [23], TSVM by gradient descent (∇TSVM) [23], low density separation (LDS) [23], Laplacian support vector machines (LapSVM) [99] and Laplacian regularized least squares (LapRLS) [99]. For WebKB, we use the PRBEP metric when comparing against these methods. The results for all older methods, except LapSVM and LapRLF, are obtained from [23], the results for LapSVM and LapRLF are from [99]. All comparisons with older methods use the same number of labeled samples as the proposed methods: 12 labels for WebKB and the PRBEP metric, and 50 labels for the rest of the data.

4.5 Computation analysis

This section is dedicated to the computational analysis of the proposed MMBO and MML methods. In particular, we elaborate on the multiscale coefficient analysis, the effect of labeled data, the computational complexity, the convergence and the efficiency of the proposed algorithms.

4.5.1 Multiscale coefficient analysis

In this subsection, we computationally determine the effect and the meaningful range of the multiscale coefficient c_t part of the multiscale graph Laplacian (12). Theoretically, the $(m+1)$-kernel method will degenerate to the m-kernel model when c_m approaches zero. Therefore, a stable optimizer should preserve that property. To this end, we examine the relative difference in the accuracy of the two-kernel and one-kernel MML algorithms over five discussed datasets:

$$\%\text{Difference} = \left| \frac{\text{Acc}_{\text{MML}}^{\text{2-kernel}} - \text{Acc}_{\text{MML}}^{\text{1-kernel}}}{\text{Acc}_{\text{MML}}^{\text{1-kernel}}} \right| \times 100\%, \quad (18)$$

where $\text{Acc}_{\text{MML}}^{\text{k-kernel}}$ indicates the classification accuracy obtained by the MML algorithm with k kernels. It is noted that the 2-kernel and 1-kernel MML methods will share the same first multiscale coefficient c_0, while the second multiscale coefficient c_1 of 2-kernel MML model is chosen from the following set: $\{10^{-3}, 0.5 \times 10^{-3}, 10^{-2}, \ldots, 10^{3}\}$.

Figure 5 illustrates the %Difference Performances between the two MML models on the five datasets. Note that we only consider the classification accuracy for the WebKB (page+link) experiment; we find that using the PRBEP metric yields a similar trend. It is expected that the difference between the performances of multi-kernel models become larger when the multiscale coefficient c_1 increases. When c_1 is smaller than 10^{-2}, the two-kernel MML algorithm performs similarly to the one-kernel counterpart, as expected. A similar conclusion can be drawn when a higher number of kernels is used in our models, and when the MMBO algorithm is utilized in the experiment.

4.5.2 Effect of labeled data

In this experiment, we examine the role of the number of labeled data elements on the accuracy of our proposed algorithms. The proposed MML and MMBO methods are semi-supervised learning approaches, which perform well with small amounts of labeled samples. Thus, our models usually do not require abundantly available labeled data. Nevertheless, we perform experiments on MMBO and MML with various amounts of labeled data.

To this end, we fix the test data for each dataset, and train our models on the rest. See Table 2 for the details of these

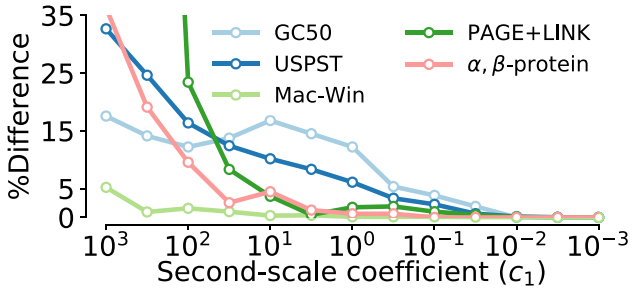

Fig. 5 The %Difference between the one-kernel and two-kernel MML models, depending on c_1. While the two models share the same multiscale coefficient c_0, the second kernel coefficient c_1 is discretely chosen in the domain $\{10^{-3}, \ldots, 10^{3}\}$. Similar behavior can be observed if the MMBO method is used instead in the experiment

settings. The number of labeled samples is determined by the percentage $d(\%)$ of the size of the training set in each experiment. Here, we vary d from 10% to 50%, with an increment of 10%. To obtain a fair comparison, we optimize both MML and MMBO hyperparameters for each amount of labeled samples. The accuracy metric is employed for both MML and MMBO performance evaluation, except for the WebKB data set, where the PRBEP metric is preferred due to the imbalance in the class distribution.

Figure 6 plots the performances of both MML and MMBO versus the different amounts of labeled samples for all five datasets. As seen in Fig. 6, for the G50C and WebKB data sets, one can obtain a very similar accuracy (differing by less than 1%) when considering 10% of the training data as labeled, as opposed to considering 50% of the training data as labeled. In addition, for the Mac-Win and USPST data sets, one can also obtain a very similar accuracy (differing by less than 5%) when considering 10% of the training data as labeled, as opposed to considering 50% of the training data as labeled These results confirm that our semi-supervised models are very accurate in the case of a small number of labeled samples. For the biological data, there is a larger change in the accuracy depending on the number of labeled samples. The complexity of the protein dataset and the quality of the features with essential physical and chemical information in the biological structures have caused the dependency of our semi-supervised learners on the knowledge of the labeled samples. This speculation will deserve a more in-depth investigation in our future work.

4.5.3 Computational complexity

In this section, we elaborate on the computational complexity of the proposed methods. In regarding to computational complexity of the MMBO algorithm, in practice, once the N_e eigenvectors of the graph Laplacian are computed, the complexity of MMBO scheme is linear in the number of data elements N. In particular, let K be the number of

Table 2 Settings for the experiments of the effect of labeled data on MML and MMBO's performances

Data set	No. of classes	Sample dim.	No. of data elements	No. of test elements
G50C	2	50	550	250
USPST	20	256	2007	979
Mac-Win	2	7511	1946	948
WebKB (page+link)	2	4840	1051	520
α, β-protein	3	50	900	180

classes and $m+1$ be the number of terms in the multiscale Laplacian (12). Usually, $m = 1$ or $m = 2$ is enough to obtain good accuracy. Then, one needs $O(NKN_e)$ operations for the multiscale heat equation with a forcing term, $O(NK\log K)$ operations for the projection to the simplex and $O(NK)$ operations for the displacement step. Moreover, the technique in [10, 33, 34] allows one to compute good approximations to the first N_e eigenvectors of the multiscale graph Laplacian using $O(NN_e(m+1))$ operations. Since $N_e << N$ and $K << N$, in practice, the complexity of this method is linear.

In regarding to computational complexity of the MML algorithm, the method uses a direct approach that requires $O(N^2)$ operations to calculate a single data-dependent kernel $\tilde{M}$ [91]. When $m+1$ kernels are utilized, the complexity in space of the MML method is $O(N^2(m+1))$. Thus, MMBO technique theoretically requires less number of operations than its counterpart, the MML procedure.

4.5.4 Convergence analysis

In this section, we study a Lyapunov function of a scheme related to fidelity forced MBO-type schemes, where labeled data is incorporated into the method. We specifically consider the two-class case. The theory can be extended to multiclass cases, as detailed in [37, 47, 89]. The groundwork

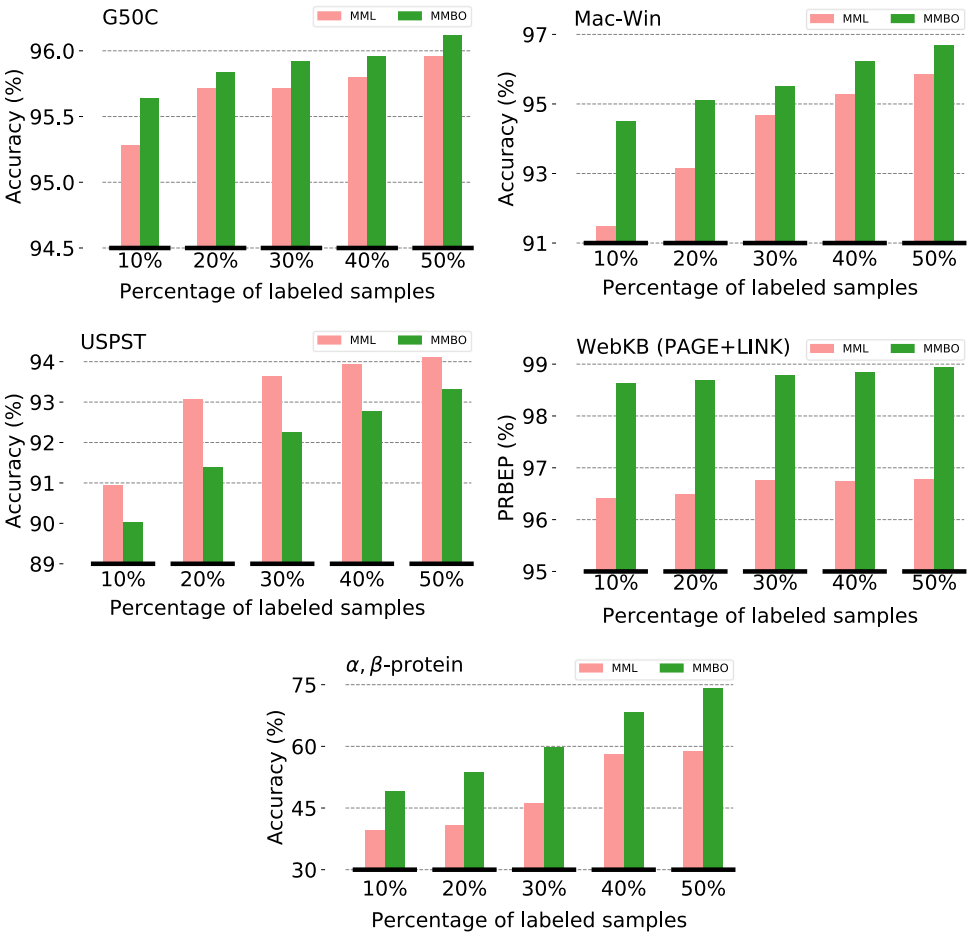

Fig. 6 The effect of the number of labeled samples on the performances of the proposed MML and MMBO methods

Table 3 The timing of the proposed MMBO method

Data set	Size of data set	Sample dimension	Timing (Construction of graph and eigenvectors)	Timing (MMBO procedure)
G50C	550	50	0.02 seconds	0.31 seconds
USPST	1440	1024	1.41 seconds	1.52 seconds
Mac-Win	1946	7511	9.8 seconds	1.17 seconds
WebKB (page)	1051	3000	1.04 seconds	0.60 seconds
WebKB (link)	1051	1840	0.67 seconds	0.60 seconds
WebKB (page+link)	1051	4840	1.58 seconds	0.60 seconds
α, β-protein	900	50	0.18 seconds	1.96 seconds

for the extension was considered in [14]. In the binary case, we can consider $\mathbf{U}$ as a vector where thresholding at $\frac{1}{2}$ divides elements into two classes. Moreover, let

$$e^{-t\mathbf{A}} = \sum_{k=0}^{\infty} (-t)^k \frac{\mathbf{A}^k}{k!}, \tag{19}$$

for any matrix $\mathbf{A}$, let $\mathbf{L}$ be the graph Laplacian (the standard, symmetric or multiscale version), $\mathbf{I}$ be the identity matrix, and $\mathbf{M}$ be a diagonal matrix with μ as its diagonal value for labeled elements, and 0 as the diagonal value otherwise, and let $\tilde{f}$ indicate the reference which is supported on the labeled data elements, and let $f = \mathbf{M}\tilde{f}$. In addition, for $t > 0$, let

$$J(\mathbf{U}) = \langle \mathbf{U}, 1 - 2(\mathbf{L}+\mathbf{M})^{-1}(\mathbf{I} - e^{-t(\mathbf{L}+\mathbf{M})})f - e^{-t(\mathbf{L}+\mathbf{M})}\mathbf{U} \rangle, \tag{20}$$

where $\langle\rangle$ indicates the dot product. As mentioned before, in the two-class case, we can consider $\mathbf{U}$ as a vector where thresholding at $\frac{1}{2}$ can divide elements into two classes.

The work [15] shows that (20) is not only strictly concave, but it is also a Lyapunov functional for a certain SDIE scheme [15], that is, $J(\mathbf{U}^{n+1}) \leq J(\mathbf{U}^n)$, with equality if and only if $\mathbf{U}^{n+1} = \mathbf{U}^n$ for $\mathbf{U}^{n+1}$ defined by the SDIE scheme. It is now very important to note that it can be shown, using elements of Theorem 2.9 from [15], that $\mathbf{U}^{n+1}$ is a solution to the SDIE scheme (with $\lambda = 1$) if and only if it solves the variational problem that defines the fidelity-forced MBO scheme, i.e., the minimization of the graph Ginzburg-Landau functional (11) plus a specific fidelity term involving an L_2 fit to labeled data. We refer the reader to [15] for other connections between MBO-type methods and the SDIE scheme.

4.5.5 Efficiency

The efficiency of the proposed MML and MMBO procedures are examined in this subsection. The timing results are listed for all data sets in Table 3 (for the MMBO algorithm) and Table 4 (for the MML algorithm).

The timing of the proposed MMBO method is divided into two parts: (1) the timing for the construction of the graph weights and the calculation of the extremal eigenvectors of the multiscale graph Laplacian, and (2) the timing of the MMBO procedure. From Table 2, one can see that the proposed MMBO procedure takes under 2 seconds for all data sets, and the graph construction and computation of the eigenvectors takes little time as well.

The timing of the proposed MML method consists of two categories: (1) the timing for the construction of the warped kernels, and (2) the timing of the optimizer. One can see from Table 4 that the procedure of generating the multiscale graph and the warped kernel is the most time-consuming step of the MML algorithm, but it is still under 5 seconds when handling the Mac-Win data set having 1946 samples with a feature dimension of 7511. For other data sets, the MML method takes under 0.3 seconds to formulate the multiscale graph and the warped kernel. Due

Table 4 The timing of the proposed MML method

Data set	Size of data set	Sample dimension	Timing (Deformed Kernel)	Timing (Optimization)
G50C	550	50	0.039 seconds	0.001 seconds
USPST	1440	1024	0.24 seconds	0.003 seconds
Mac-Win	1946	7511	4.51 seconds	0.002 seconds
WebKB (page)	1051	3000	0.21 seconds	0.02 seconds
WebKB (link)	1051	1840	0.15 seconds	0.01 seconds
WebKB (page+link)	1051	4840	0.28 seconds	0.02 seconds
α, β-protein	900	50	0.05 seconds	0.02 seconds

to the simplified version of the optimizer of the MML method, one can directly use the standard solver of SVM for the MML algorithm. This procedure is extremely fast and needs no more than 0.03 seconds to complete the task for all experiments. The computations were performed on a personal laptop 2.4 GHz 8-Core Intel Core i9.

5 Conclusion

This work presents several methods for machine learning tasks and for dealing with some of the challenges of machine learning, such as data with limited samples, smaller data sets, and diverse data, usually associated with small data sets or data related to areas of study where the size of the data sets is constrained by the complexity and/or high cost of experiments. In particular, we integrate graph-based techniques, multiscale structure, adapted and modified optimization procedures and semi-supervised frameworks to derive two multiscale Laplacian learning (MLL) approaches for machine learning tasks, such as data classification.

The first approach introduces a multiscale kernel representation to a manifold learning technique and is called the multikernel manifold learning (MML) algorithm. The second approach combines multiscale analysis with an interesting adaptation and modification of the famous classical Merriman-Bence-Osher (MBO) scheme and is called the multiscale MBO (MMBO) algorithm.

The performance of the proposed approaches is favorably compared to recent and related approaches through experiments on various data sets. A variety of computational analyses indicates that two proposed new MLL methods are powerful techniques for dealing with some of the most important challenges and tasks in machine learning.

6 Supporting information

We present the optimal hyperparameters of the proposed methods in Online Resource: Supporting Information.

Acknowledgements This work was supported in part by NSF Grants DMS-2052983, DMS-2053284, DMS2151802, DMS-1761320, and IIS1900473, NIH grants R01GM126189 and NIH R01AI164266, Bristol-Myers Squibb, Pfizer, MSU Foundation, and University of Kentucky Startup Fund.

Code Availability The source code is available at Github: https://github.com/ddnguyenmath/Multiscale-Laplacian-Learning.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Scholkopf B, Herbrich R, Smola AJ (2001) A Generalized Representer Theorem. In: 14th Annual conference on computational learning theory. https://alex.smola.org/papers/2001/SchHerSmo01.pdf
2. Huilgol P (2020) Quick introduction to bag-of-words (BoW) and TF-IDF for creating features from text. https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/
3. Vedaldi A, Fulkerson B (2008). VLFeat Library. https://www.vlfeat.org
4. Abu-El-Haija S, Kapoor A, Perozzi B, Lee J (2018) N-GCN: multi-scale graph convolution for semi-supervised node classification. arXiv:1802.08888
5. Anam K, Al-Jumaily A (2015) A novel extreme learning machine for dimensionality reduction on finger movement classification using sEMG. In: International IEEE/EMBS conference on neural engineering. IEEE, pp 824–827
6. Anderson C (2010) A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices. J Comput Phys 229:7477–7487
7. Bae E, Merkurjev E (2017) Convex variational methods on graphs for multiclass segmentation of high-dimensional data and point clouds. J Math Imaging Vis 58(3):468–493
8. Belkin M, Matveeva I, Niyogi P (2004) Regularization and semi-supervised learning on large graphs. In: International conference on computational learning theory. Springer, pp 624–638
9. Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7(Nov):2399–2434
10. Belongie S, Fowlkes C, Chung F, Malik J (2002) Spectral partitioning with indefinite kernels using the nyström extension. In: European conference on computer vision. Springer, pp 531–542
11. Bohacek RS, McMartin C, Guida WC (1996) The art and practice of structure-based drug design: a molecular modeling perspective. Med Res Rev 16(1):3–50
12. Boykov Y, Veksler O, Zabih R (1999) Fast approximate energy minimization via graph cuts. In: ICCV (1), pp 377–384. citeseer.ist.psu.edu/boykov99fast.html
13. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. arXiv:1312.6203
14. Budd J, van Gennip Y (2020) Graph merriman-bence-osher as a semi-discrete implicit euler scheme for graph allen-cahn flow. SIAM J Math Anal 52(5):4101–4139
15. Budd J, van Gennip Y, Latz J (2021) Classification and image processing with a semi-discrete scheme for fidelity forced allen–cahn on graphs. Gesellschaft fur Angewandte Mathematik und Mechanik 44(1):e202100004
16. Cang Z, Mu L, Wei GW (2018) Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. PLos Computat Bio 14(1):e1005929
17. Cang ZX, Mu L, Wu K, Opron K, Xia K, Wei GW (2015) A topological approach to protein classification. Molecular Based Math Bio 3:140–162
18. Cang ZX, Wei GW (2017) Analysis and prediction of protein folding energy changes upon mutation by element specific persistent homology. Bioinformatics 33:3549–3557
19. Cang ZX, Wei GW (2017) TopologyNet: topology based deep convolutional and multi-task neural networks for biomolecular property predictions. PLOS Computat Bio 13(7):e1005690. https://doi.org/10.1371/journal.pcbi.1005690
20. Cang ZX, Wei GW (2018) Integration of element specific persistent homology and machine learning for protein-ligand

binding affinity prediction. Int J Numer Methods Biomed Eng, vol 34(2). https://doi.org/10.1002/cnm.2914
21. Cevikalp H, Franc V (2017) Large-scale robust transductive support vector machines. Neurocomputing 235:199–209
22. Chapelle O, Schölkopf B, Zien A (2006) Semi-supervised learning. MIT Press, Cambridge, MA
23. Chapelle O, Zien A (2005) Semi-supervised classification by low density separation. In: AISTATS. Citeseer, vol 2005, pp 57–64
24. Chen C, Xin J, Wang Y, Chen L, Ng MK (2018) A semisupervised classification approach for multidomain networks with domain selection. IEEE Trans Neural Netw Learn Syst 30(1):269–283
25. Chen J, Zhao R, Tong Y, Wei GW, Vedaldi A., Fulkerson B. (2021) Evolutionary de rham-hodge method. Discrete & continuous dynamical systems - B (In press, 2020)
26. Chen Y, Ye X (2011) Projection onto a simplex. arXiv:1101.6081
27. Cheng Y, Zhao X, Cai R, Li Z, Huang K, Rui Y (2016) Semi-supervised multimodal deep learning for RGB-d object recognition. In: International joint conferences on artificial intelligence, pp 3345–3351
28. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
29. Couprie C, Grady L, Najman L, Talbot H (2011) Power watershed: a unifying graph-based optimization framework. IEEE Trans Pattern Anal Mach Intell 33(7):1384–1399
30. Elmoataz A, Lezoray O, Bougleux S (2008) Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. IEEE Trans Image Process 17(7):1047–1060
31. Fang X, Xu Y, Li X, Fan Z, Liu H, Chen Y (2014) Locality and similarity preserving embedding for feature selection. Neurocomputing 128:304–315
32. Feng S, Zhou H, Dong H (2019) Using deep neural network with small dataset to predict material defects. Mater Des 162:300–310
33. Fowlkes C, Belongie S, Chung F, Malik J (2004) Spectral grouping using the Nyström method. IEEE Trans Pattern Anal Mach Intell 26(2):214–225
34. Fowlkes C, Belongie S, Malik J (2001) Efficient spatiotemporal grouping using the Nyström method. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. IEEE, vol 1, pp I–I
35. Fox NK, Brenner SE, Chandonia JM (2014) SCOPe: structural classification of proteins-extended, integrating SCOP and ASTRAL data and classification of new structures. Nucleic Acids Res 42(D1):D304–D309
36. Gadde A, Anis A, Ortega A (2014) Active semi-supervised learning using sampling theory for graph signals. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 492–501
37. Garcia-Cardona C, Merkurjev E, Bertozzi AL, Flenner A, Percus A (2014) Fast multiclass segmentation using diffuse interface methods on graphs. IEEE Trans Pattern Anal Mach Intell
38. Gerhart T, Sunu J, Lieu L, Merkurjev E, Chang JM, Gilles J, Bertozzi AL (2013) Detection and tracking of gas plumes in LWIR hyperspectral video sequence data. In: SPIE conference on defense, security, and sensing, pp 87430j–87430j
39. Goldberg AB, Zhu X, Wright S (2007) Dissimilarity in graph-based semi-supervised classification. In: Artificial intelligence and statistics, pp 155–162
40. Gong C, Tao D, Maybank S, Liu W, Kang G, Yang J (2016) Multi-modal curriculum learning for semi-supervised image classification. IEEE Trans Image Process 25(7):3249–3260
41. Grandvalet Y, Bengio Y (2005) Semi-supervised learning by entropy minimization. In: Advances in neural information processing systems, pp 529–536
42. Grimes RG, Lewis JG, Simon HD (1994) A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. SIAM J Matrix Anal Appl 15(1):228–272
43. Huang G, Song S, Gupta J, Wu C (2014) Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern 44(12):2405–2417
44. Huang G, Song S, Xu ZE, Weinberger K (2014) Transductive minimax probability machine. In: Joint european conference on machine learning and knowledge discovery in databases. Springer, pp 579–594
45. Hudson DL, Cohen ME (2000) Neural networks and artificial intelligence for biomedical engineering. Institute Electr Electron Eng
46. Iscen A, Tolias G, Avrithis Y, Chum O (2019) Label propagation for deep semi-supervised learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5070–5079
47. Jacobs M, Merkurjev E, Esedoğlu S (2018) Auction dynamics: a volume constrained mbo scheme. J Comput Phys 354:288–310
48. Jia X, Jing XY, Zhu X, Chen S, Du B, Cai Z, He Z, Yue D (2020) Semi-supervised multi-view deep discriminant representation learning. IEEE Trans Pattern Anal Mach Intell
49. Jiang J, Wang R, Wang M, Gao K, Nguyen DD, Wei GW (2020) Boosting tree-assisted multitask deep learning for small scientific datasets. J Chem Inf Model 60(3):1235–1244
50. Joachims T et al (1999) Transductive inference for text classification using support vector machines. In: International conference on machine learning, vol 99, pp 200–209
51. Jordan MI, Mitchell TM (2015) Machine learning: trends, perspectives, and prospects. Science 349(6245):255–260
52. Kapoor A, Ahn H, Qi Y, Picard RW (2006) Hyperparameter and kernel learning for graph based semi-supervised classification. In: Advances in neural information processing systems, pp 627–634
53. Kasun L, Yang Y, Huang GB, Zhang Z (2016) Dimension reduction with extreme learning machine. IEEE Trans Image Process 25(8):3906–3918
54. Katz G, Caragea C, Shabtai A (2017) Vertical ensemble co-training for text classification. ACM Trans Intell Syst Technol 9(2):1–23
55. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
56. Kotsiantis SB, Zaharakis I, Pintelas P (2007) Supervised machine learning: a review of classification techniques. Emerging Artif Intell Appl Comput Eng 160(1):3–24
57. Levin A, Rav-Acha A, Lischinski D (2008) Spectral matting. IEEE Trans Pattern Anal Mach Intell 30(10):1699–1712
58. Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-second AAAI conference on artificial intelligence
59. Li YF, Zhou ZH (2014) Towards making unlabeled data never hurt. IEEE Trans Pattern Anal Mach Intell 37(1):175–188
60. Liao R, Brockschmidt M, Tarlow D, Gaunt A, Urtasun R, Zemel RS (2018) Graph partition neural networks for semi-supervised classification. https://openreview.net/forum?id=rk4Fz2e0b
61. Lin F, Cohen WW (2010) Semi-supervised classification of network data using very few labels. In: 2010 International conference on advances in social networks analysis and mining. IEEE, pp 192–199
62. Liu Y, Ng MK, Zhu H (2021) Multiple graph semi-supervised clustering with automatic calculation of graph associations. Neurocomputing 429:33–46

63. Luo T, Hou C, Nie F, Yi D (2018) Dimension reduction for non-Gaussian data by adaptive discriminative analysis. IEEE Trans Cybern 49(3):933–946
64. Melacci S, Belkin M (2011) Laplacian support vector machines trained in the primal. J Mach Learn Res, vol 12(3)
65. Meng G, Merkurjev E, Koniges A, Bertozzi AL (2017) Hyperspectral video analysis using graph clustering methods. Image Process Line 7:218–245
66. Merkurjev E, Bertozzi AL, Chung F (2018) A semi-supervised heat kernel pagerank mbo algorithm for data classification. Commun Math Sci 16(5):1241–1265
67. Merkurjev E, Bertozzi AL, Lerman K, Yan X (2017) Modified Cheeger and ratio cut methods using the Ginzburg-Landau functional for classification of high-dimensional data. Inverse Probl 33(7):074003
68. Merkurjev E, Garcia-Cardona C, Bertozzi AL, Flenner A, Percus A (2014) Diffuse interface methods for multiclass segmentation of high-dimensional data. Appl Math Lett 33:29–34
69. Merkurjev E, Kostic T, Bertozzi AL (2013) An MBO scheme on graphs for segmentation and image processing. SIAM J Imaging Sci 6(4):1903–1930
70. Merkurjev E, Sunu J, Bertozzi AL (2014) Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video. Proc IEEE Int Conf Image Process
71. Merriman B, Bence JK, Osher S (1992) Diffusion generated motion by mean curvature. AMS Select Lect Math Series Computat Crystal Growers Workshop 8966:73–83
72. Merriman B, Bence JK, Osher SJ (1994) Motion of multiple functions: a level set approach. J Computat Phys 112(2):334–363. https://doi.org/10.1006/jcph.1994.1105
73. Nguyen D, Wei GW (2019) Agl-score: algebraic graph learning score for protein-ligand binding scoring, ranking, docking, and screening. J Chem Inf Model
74. Nguyen DD, Cang Z, Wei GW (2020) A review of mathematical representations of biomolecular data. Phys Chem Chem Phys 22(8):4343–4367
75. Nguyen DD, Wei GW (2019) DG-GL: differential Geometry-based geometric learning of molecular datasets. Int J Numer Methods Biomed Eng 35(3):e3179
76. Nguyen DD, Xia K, Wei GW (2016) Generalized flexibility-rigidity index. J Chem Phys 144(23):234106
77. Nguyen DD, Xiao T, Wang ML, Wei GW (2017) Rigidity strengthening: a mechanism for protein-ligand binding. J Chem Inf Model 57:1715–1721
78. Ni T, Chung FL, Wang S (2015) Support vector machine with manifold regularization and partially labeling privacy protection. Inf Sci 294:390–407
79. Nie F, Cai G, Li J, Li X (2017) Auto-weighted multi-view learning for image clustering and semi-supervised classification. IEEE Trans Image Process 27(3):1501–1511
80. Nie F, Cai G, Li X (2017) Multi-view clustering and semi-supervised classification with adaptive neighbours. In: Thirty-first AAAI conference on artificial intelligence
81. Nie F, Li J, Li X (2016) Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification. In: IJCAI, pp 1881–1887
82. Nie F, Li J, Li X (2016) Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification. In: International joint conferences on artificial intelligence, pp 1881–1887
83. Nie F, Wang X, Huang H (2014) Clustering and projected clustering with adaptive neighbors. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 977–986
84. Noroozi V, Bahaadini S, Zheng L, Xie S, Shao W, Philip SY (2018) Semi-supervised deep representation learning for multi-view problems. In: IEEE international conference on big data. IEEE, pp 56–64
85. Opron K, Xia K, Wei GW (2015) Communication: capturing protein multiscale thermal fluctuations
86. Paige CC (1972) Computational variants of the Lanczos method for the eigenproblem. IMA J Appl Math 10(3):373–381
87. Perona P, Zelnik-Manor L (2004) Self-tuning spectral clustering. Adv Neural Inf Process Syst
88. Qi Z, Tian Y, Shi Y (2012) Laplacian twin support vector machine for semi-supervised classification. Neural Netw 35:46–53
89. Qiao Y, Shi C, Wang C, Li H, Haberland M, Luo X, Stuart AM, Bertozzi AL (2019) Uncertainty quantification for semi-supervised multi-class classification in image processing and ego-motion analysis of body-worn videos. Electron Imaging 2019(11):264–1
90. Qu M, Bengio Y, Tang J (2019) GMNN: graph Markov neural networks. arXiv:1905.06214
91. Rudi A, Carratino L, Rosasco L (2017) Falkon: an optimal large scale kernel method. Adv Neural Inf Process Syst, vol 30
92. Saha B, Gupta S, Phung D, Venkatesh S (2016) Multiple task transfer learning with small sample sizes. Knowl Inf Syst 46(2):315–342
93. Sakai T, Plessis MC, Niu G, Sugiyama M (2017) Semi-supervised classification based on classification from positive and unlabeled data. In: International conference on machine learning. PMLR, pp 2998–3006
94. Sansone E, Passerini A, De Natale F (2016) Classtering: joint classification and clustering with mixture of factor analysers. In: Proceedings of the twenty-second european conference on artificial intelligence, pp 1089–1095
95. Schwab K (2017) The fourth industrial revolution currency
96. Shaikhina T, Khovanova NA (2017) Handling limited datasets with neural networks in medical applications: a small-data approach. Artif Intell Med 75:51–63
97. Shaikhina T, Lowe D, Daga S, Briggs D, Higgins R, Khovanova N (2015) Machine learning for predictive modelling based on small data in biomedical engineering. IFAC-PapersOnLine 48(20):469–474
98. She Q, Hu B, Luo Z, Nguyen T, Zhang Y (2019) A hierarchical semi-supervised extreme learning machine method for EEG recognition. Med Biol Eng Comput 57(1):147–157
99. Sindhwani V, Niyogi P, Belkin M (2005) Beyond the point cloud: from transductive to semi-supervised learning. In: Proceedings of the 22nd international conference on machine learning. ACM, pp 824–831
100. Subramanya A, Bilmes J (2011) Semi-supervised learning with measure propagation. J Mach Learn Res 12:3311–3370
101. Szummer M, Jaakkola T (2002) Partially labeled classification with markov random walks. In: Advances in neural information processing systems, pp 945–952
102. Thekumparampil KK, Wang C, Oh S, Li LJ (2018) Attention-based graph neural network for semi-supervised learning. arXiv:1803.03735
103. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
104. Wang B, Tu Z, Tsotsos JK (2013) Dynamic label propagation for semi-supervised multi-class multi-label classification. In: Proceedings of the IEEE international conference on computer vision, pp 425–432
105. Wang J, Jebara T, Chang SF (2013) Semi-supervised learning using greedy max-cut. J Mach Learn Res 14(Mar):771–800

106. Wang M, Fu W, Hao S, Tao D, Wu X (2016) Scalable semi-supervised learning by efficient anchor graph regularization. IEEE Trans Knowl Data Eng 28(7):1864–1877
107. Wang Q, Qin Z, Nie F, Li X (2020) C2dnda: a deep framework for nonlinear dimensionality reduction. IEEE Trans Ind Electron 68(2):1684–1694
108. Wang R, Nguyen DD, Wei GW (2020) Persistent spectral graph. Int J Numer Methods Biomed Eng:e3376
109. Wang W, Arora R, Livescu K, Bilmes J (2015) On deep multi-view representation learning. In: International conference on machine learning. PMLR, pp 1083–1092
110. Weston J, Ratle F, Mobahi H, Collobert R (2012) Deep learning via semi-supervised embedding. In: Neural networks: tricks of the trade. Springer, pp 639–655
111. Xia K, Opron K, Wei GW (2015) Multiscale Gaussian network model (mGNM) and multiscale anisotropic network model (mANM). J Chem Phys 143(20):11B616_1
112. Yang L, Song S, Li S, Chen Y, Huang G (2019) Graph embedding-based dimension reduction with extreme learning machine. IEEE Trans Syst Man Cybern Syst
113. Yang Y, Wu QJ, Wang Y (2016) Autoencoder with invertible functions for dimension reduction and image reconstruction. IEEE Trans Syst Man Cybern Syst 48(7):1065–1079
114. Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning, pp 40–48
115. Zhang B, Qiang Q, Wang F, Nie F (2020) Fast multi-view semi-supervised learning with learned graph. IEEE Trans Knowl Data Eng
116. Zhang Y, Ling C (2018) A strategy to apply machine learning to small datasets in materials science. Npj Computat Materials 4(1):1–8
117. Zhang Y, Pal S, Coates M, Ustebay D (2019) Bayesian graph convolutional neural networks for semi-supervised classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 5829–5836
118. Zhao H, Zheng J, Deng W, Song Y (2020) Semi-supervised broad learning system based on manifold regularization and broad network. IEEE Trans Circuits Syst Regular Papers 67(3):983–994
119. Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in neural information processing systems 16. MIT Press, Cambridge, MA, pp 321-328
120. Zhou D, Schölkopf B (2004) A regularization framework for learning from graph data. In: Workshop on statistical relational learning. International conference on machine learning, Banff, Canada
121. Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation. CMU CALD Tech Report CMU-CALD-02-107
122. Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning, pp 912–919
123. Zhuang C, Ma Q (2018) Dual graph convolutional networks for graph-based semi-supervised classification. In: Proceedings of the 2018 world wide web conference, pp 499–508