Mach Learn (2017) 106:307–335
DOI 10.1007/s10994-016-5607-3

Adaptive edge weighting for graph-based learning algorithms

Masayuki Karasuyama[1] · **Hiroshi Mamitsuka**[2,3]

Received: 9 September 2015 / Accepted: 1 November 2016 / Published online: 18 November 2016

Abstract Graph-based learning algorithms including label propagation and spectral clustering are known as the effective state-of-the-art algorithms for a variety of tasks in machine learning applications. Given input data, i.e. feature vectors, graph-based methods typically proceed with the following three steps: (1) generating graph edges, (2) estimating edge weights and (3) running a graph based algorithm. The first and second steps are difficult, especially when there are only a few (or no) labeled instances, while they are important because the performance of graph-based methods heavily depends on the quality of the input graph. For the second step of the three-step procedure, we propose a new method, which optimizes edge weights through a local linear reconstruction error minimization under a constraint that edges are parameterized by a similarity function of node pairs. As a result our generated graph can capture the manifold structure of the input data, where each edge represents similarity of each node pair. To further justify this approach, we also provide analytical considerations for our formulation such as an interpretation as a cross-validation of a propagation model in the feature space, and an error analysis based on a low dimensional manifold model. Experimental results demonstrated the effectiveness of our adaptive edge weighting strategy both in synthetic and real datasets.

Keywords Graph-based learning · Manifold assumption · Edge weighting · Semi-supervised learning · Clustering

Editor: Karsten Borgwardt.

✉ Masayuki Karasuyama
karasuyama@nitech.ac.jp

Hiroshi Mamitsuka
mami@kuicr.kyoto-u.ac.jp

[1] Department of Engineering, Nagoya Institute of Technology, Gokiso, Showa-ku, Nagoya, Aichi 466-8555, Japan

[2] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasyo, Uji, Kyoto 611-0011, Japan

[3] Department of Computer Science, Aalto University, 02150 Espoo, Finland

1 Introduction

Graph-based learning algorithms have received considerable attention in machine learning community. For example, *label propagation* (e.g., Blum and Chawla 2001; Szummer and Jaakkola 2001; Joachims 2003; Zhu et al. 2003; Zhou et al. 2004; Herbster et al. 2005; Sindhwani et al. 2005; Belkin et al. 2006; Bengio et al. 2006) is widely accepted as a state-of-the-art approach for semi-supervised learning, in which node labels are estimated through the input graph structure. *Spectral clustering* (e.g., Shi and Malik 2000; Ng et al. 2001; Meila and Shi 2001; von Luxburg 2007) is also a famous graph-based algorithm, in which cluster partitions are determined according to the minimum cut of the given graph. A common important property of these graph-based approaches is that the *manifold* structure of the input data can be captured by the graph. Their practical performance advantage has been demonstrated in various application areas (e.g., Patwari and Hero 2004; Lee and Kriegman 2005; Zhang and Zha 2005; Fergus et al. 2009; Aljabar et al. 2012).

On the other hand, it is well-known that the accuracy of the graph-based methods highly depends on the quality of the input graph (e.g., Zhu et al. 2003; Kapoor et al. 2006; Zhang and Lee 2007; von Luxburg 2007; Wang and Zhang 2008), which is typically generated from a set of numerical input vectors (i.e., feature vectors). A general framework of graph-based learning can be represented as the following *three-step procedure*:

Step 1: Generating graph edges from given data, where nodes of the generated graph correspond to the instances of input data.
Step 2: Giving weights to the graph edges.
Step 3: Estimating node labels based on the generated graph, which is often represented as an *adjacency matrix*.

This framework is employed by many graph-based algorithms including label propagation and spectral clustering.

In this paper, we focus on the second step in the three-step procedure; estimating edge weights for the subsequent label estimation. Optimizing edge weights is difficult in semi- or un-supervised learning, because there are only a small number of (or no) labeled instances. Also this problem is important because edge weights heavily affect the final prediction accuracy of graph-based methods, while in reality rather simple heuristics strategies have been employed.

There are two standard approaches for estimating edge weights: similarity function based- and *locally linear embedding* (LLE) (Roweis and Saul 2000) based-approaches. Each of these two approaches has its own disadvantage. The similarity based approaches use similarity functions, such as Gaussian kernel, while most similarity functions have scale parameters (such as the width parameter of Gaussian kernel) that are in general difficult to be tuned. On the other hand, in LLE, the true underlying manifold can be approximated by a graph by minimizing a local reconstruction error. LLE is more sophisticated than the similarity-based approach, and LLE based graphs have been applied to semi-supervised learning and clustering (Wang and Zhang 2008; Daitch et al. 2009; Cheng et al. 2009; Liu et al. 2010). However LLE is noise-sensitive (Chen and Liu 2011). In addition, to avoid a kind of degeneracy problem (Saul and Roweis 2003), LLE has to have additional tuning parameters.[1] Yet another practical approach is to optimize weights by regarding them as *hyper-parameters* of learning methods (e.g., Zhang and Lee 2007). Also general model selection criteria can

[1] The LLE based approaches can be interpreted as simultaneously performing the first two steps in the three-step procedure because both edges and weights are obtained.

be used, while the reliability of those criteria are unclear for graphs with a small number of labeled instances. We will discuss those related approaches in Sect. 5.

Our approach is a similarity-based method, yet also captures the manifold structure of the input data; we refer to our approach as *adaptive edge weighting* (AEW) because graph edges are determined by a data adaptive manner in terms of both similarity and manifold structure. The objective function in AEW is based on local reconstruction, by which estimated weights capture the manifold structure. However, unlike conventional LLE based approaches, we introduce an additional constraint that edges represent similarity of two nodes. Due to this constraint, AEW has the following advantages compared to LLE based approaches:

- Our formulation alleviates the problem of over-fitting due to the parameterization of weights. We observed that AEW is more robust against noise of input data and the change of the number of graph edges.
- Since edge weights are defined as a parameterized similarity function, resultant weights still represent the similarity of each node pair. This is very reasonable for many graph-based algorithms.

We provide further justifications for our approach based on the ideas of *feature propagation* and *local linear approximation*. Our objective function can be seen as a cross validation error of a propagation model for feature vectors, which we call feature propagation. This allows us to interpret that AEW optimizes graph weights through cross validation (for prediction) in the feature vector space instead of given labels, assuming that input feature vectors and given labels share the same local structure. Another interpretation is provided through local linear approximation, by which we can analyze the error of local reconstruction in the output (label) space under the assumption of low dimensional manifold model.

The rest of this paper is organized as follows: In Sect. 2, we briefly review some standard algorithms in graph-based methods on which we focus in this paper. Section 3 introduces our proposed method for adaptively optimizing graph edge weights. Section 4 describes analytical consideration for our approach which provides interesting interpretations and error analysis of AEW. In Sect. 5, we discuss relationships to other existing topics. Section 6 presents experimental results obtained by a variety of datasets including synthetic and real-world datasets, demonstrating the performance advantage of the proposed approach. Finally, Sect. 7 concludes the paper.

This paper is an extended version of our preliminary conference paper presented at NIPS 2013 (Karasuyama and Mamitsuka 2013). In this paper, we describe our framework in a more general way by using three well-known graph-based learning methods [harmonic Gaussian field (HGF) model, local global consistency (LLGC) method, and spectral clustering], while the preliminary version only deals with HGF. Furthermore, we have conducted experimental evaluation more thoroughly which includes mainly three points: in semi-supervised setting, (1) comparison with other state-of-the-art semi-supervised methods and (2) comparison with hyper-parameter optimization methods, and as an additional problem setting, (3) comparisons on the clustering.

2 Graph-based semi-supervised learning and clustering

In this paper we consider label propagation and spectral clustering as the methods in the third step in the three-step procedure. Both are the state-of-the-art graph-based learning algorithms, and labels of graph nodes (or clusters) are estimated by using a given adjacency matrix.

Suppose that we have n feature vectors $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x} \in \mathbb{R}^p$. An undirected graph $\mathcal{G}$ is generated from $\mathcal{X}$, where each node (or vertex) corresponds to each data point $\boldsymbol{x}_i$. The graph $\mathcal{G}$ can be represented by the adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ where (i, j)-element W_{ij} is a weight of the edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The key idea of graph-based classification is that instances connected by large weights W_{ij} on a graph tend to have the same labels (meaning that labels are kept the same in the strongly connected region of the graph).

Let $\boldsymbol{F} \in \mathbb{R}^{n \times c}$ be a *label score* matrix which gives estimation of labels, where c is the number of classes or clusters. To realize the key idea described above, graph-based approaches force the label scores to have similar values for strongly connected nodes. (This corresponds to the "smoothness" penalty commonly called in semi-supervised learning literature though "continuity" is a more appropriate word to represent it.) A penalty for the variation of the score $\boldsymbol{F}$ on the graph can be represented as

$$\sum_{k=1}^{c} \sum_{ij} W_{ij} (F_{ik} - F_{jk})^2. \tag{1}$$

For the adjacency matrix W_{ij}, the following weighted *k-nearest neighbor* (k-NN) graph is commonly used in graph-based learning algorithms:

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{2\sigma^2}\right), & j \in \mathcal{N}_i \text{ or } i \in \mathcal{N}_j, \\ 0, & \text{otherwise}, \end{cases}$$

where $\mathcal{N}_i$ is a set of indices of the k-NN of $\boldsymbol{x}_i$ (Note that the adjacency matrix $\boldsymbol{W}$ is not necessarily positive definite).

From this adjacency matrix, the *graph Laplacian* (e.g., Chung 1997) can be defined by

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W},$$

where $\boldsymbol{D}$ is a diagonal matrix with the diagonal entry $D_{ii} = \sum_j W_{ij}$. Instead of $\boldsymbol{L}$, normalized variants of Laplacian such as $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}$ or $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ is also used, where $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. Using the graph Laplacian, the score variation penalty (1) can also be written as

$$\operatorname{trace}\left(\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}\right), \tag{2}$$

where trace$(\cdot)$ is defined as the sum of the diagonal entries of a given matrix.

2.1 Label propagation

Label propagation is a widely-accepted graph-based semi-supervised learning algorithm. Among many methods which have been proposed so far, we focus on the formulation derived by Zhu et al. (2003) and Zhou et al. (2004), which is the current standard formulation of graph-based semi-supervised learning.

Suppose that the first ℓ data points in $\mathcal{X}$ are labeled by $\mathcal{Y} = \{y_1, \ldots, y_\ell\}$, where $y_i \in \{1, \ldots, c\}$ and c is the number of classes. The goal of label propagation is to predict the labels of unlabeled nodes $\{\boldsymbol{x}_{\ell+1}, \ldots, \boldsymbol{x}_n\}$. The scoring matrix $\boldsymbol{F}$ gives an estimation of the label of $\boldsymbol{x}_i$ by $\operatorname{argmax}_{j \le c} F_{ij}$. Label propagation can be defined as estimating $\boldsymbol{F}$ in such a way that the score $\boldsymbol{F}$ has a smaller amount of changes on neighboring nodes as well as it can accurately predict given labeled points. The following is a regularization formulation of label propagation (Zhou et al. 2004):

$$\min_{\boldsymbol{F}} \operatorname{trace}\left(\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}\right) + \lambda \|\boldsymbol{F} - \boldsymbol{Y}\|_F^2, \tag{3}$$

where $\boldsymbol{Y} \in \mathbb{R}^{n\times c}$ is the label matrix with $Y_{ij} = 1$ if $\boldsymbol{x}_i$ is labeled as $y_i = j$; otherwise, $Y_{ij} = 0$, λ is the regularization parameter, and $\|\cdot\|_F^2$ denotes the matrix Frobenius norm defined as $\|\boldsymbol{M}\|_F^2 = \sum_{ij} M_{ij}^2$. This formulation is called *local and global consistency* (LLGC) method. The first term of LLGC (3) represents the penalty for the score differences on neighboring nodes and the second term represents the deviation from the initial labeling $\boldsymbol{Y}$. The following is another standard formulation, which is called the *harmonic Gaussian field* (HGF) model, of label propagation (Zhu et al. 2003):

$$\min_{\boldsymbol{F}} \operatorname{trace}\left(\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}\right)$$
$$\text{subject to } F_{ij} = Y_{ij}, \quad \text{for} \quad i = 1, \ldots, \ell.$$

In this formulation, the scores for labeled nodes are fixed as constants. These two formulations can be both reduced to linear systems, which can be solved efficiently, especially when Laplacian $\boldsymbol{L}$ has some sparse structure (Spielman and Teng 2004).

2.2 Spectral clustering

Spectral clustering exploits the manifold structure of input data through a given graph. The objective function can be represented as follows (von Luxburg 2007):

$$\min_{\boldsymbol{F}} \operatorname{trace}\left(\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}\right)$$
$$\text{subject to } \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}.$$

Here c, the number of columns of $\boldsymbol{F}$, is corresponding to the number of clusters which should be specified beforehand. We can solve this optimization problem through the eigenvalue decomposition of the graph Laplacian matrix, and then the partitions can be obtained by applying k-means clustering to the obtained eigenvectors. Although the formulation was originally derived as an approximation of the graph mincut problem, it can also be interpreted as minimizing the label score variation on the graph.

3 Basic framework

The performance of the graph-based algorithms, which are described in the previous section, heavily depends on the quality of the input graph. Our proposed approach, *adaptive edge weighting* (AEW), optimizes the edge weights for the graph-based learning algorithms. In the three step procedure, we note that AEW is for the second step and has nothing to do with the first and third steps. In this paper we consider that the input graph is generated by k-NN graph (the first step is based on k-NN), while we note that AEW can be applied to any types of graphs.

First of all graph edges should satisfy the following conditions:

- Capturing the manifold structure of the input space.
- Representing similarity between two nodes.

These two conditions are closely related to *manifold assumption* of graph-based learning algorithms, in which label scores should have similar values if two data points are close in the input manifold. Since the manifold structure of the input data is unknown beforehand,

the graph is used to approximate the manifold (the first condition). Subsequent predictions are performed in such a way that label scores are consistent with the similarity structure provided by the graph (the second condition). Our algorithm simultaneously pursues these two important aspects of the graph for the graph-based learning algorithms.

3.1 Formulation

We first define W_{ij} as a similarity function of two nodes (i, j), and we employ the following standard kernel function for a similarity measure (Note: other similarity functions can also be used):

$$W_{ij} = \exp\left(-\sum_{d=1}^{p} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}\right) \quad \text{for } i \in \mathcal{N}_j \text{ or } j \in \mathcal{N}_i, \tag{4}$$

where x_{id} is the dth element of $\boldsymbol{x}_i$, and $\{\sigma_d\}_{d=1}^p$ is a set of parameters. This edge weighting is commonly used in many graph-based algorithms, and this weighting can also be interpreted as the solution of the heat equation on the graph (Zhu et al. 2003). To adapt the difference of scaling in each data point, we can also use the following *local scaling kernel* (Zelnik-Manor and Perona 2004):

$$W_{ij} = \exp\left(-\sum_{d=1}^{p} \frac{(x_{id} - x_{jd})^2}{s_i s_j \sigma_d^2}\right) \quad \text{for } i \in \mathcal{N}_j \quad \text{or} \quad j \in \mathcal{N}_i, \tag{5}$$

where the constant $s_i \geq 0$ is defined as the distance to the K-th neighbor from $\boldsymbol{x}_i$ (e.g., $K = 7$ in Zelnik-Manor and Perona 2004).

To optimize parameter σ_d, we evaluate how well the graph fits the input features (manifold) by the following objective function:

$$\min_{\{\sigma_d\}_{d=1}^p} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \frac{1}{D_{ii}} \sum_{j \sim i} W_{ij} \boldsymbol{x}_j \right\|_2^2, \tag{6}$$

where $j \sim i$ means that j is connected to i. This objective function represents the local reconstruction error by local linear patch, which captures the input manifold structure (Roweis and Saul 2000; Saul and Roweis 2003). Figure 1 illustrates the idea of this approach. Similar objective functions have been used in *locally linear embedding* (LLE) (Roweis and Saul 2000) and *graph construction* (Jebara et al. 2009; Daitch et al. 2009; Talukdar 2009; Liu et al. 2010), from which the difference of our approach is that the parameters of the similarity function are optimized. The LLE type objective function assumes that a flat model provides a good local approximation of the underlying manifold (by regarding that higher-order behaviors of the manifold are not dominant locally). The parameters of a set of locally fitted flat models $\boldsymbol{W}$ are expected to reflect an intrinsic geometrical relation in a sense that the same parameters $\boldsymbol{W}$ can also approximately reconstruct each data point locally in the lower dimensional manifold by its neighbors. This property is also advantageous for the graph-based semi-supervised setting in which labels are propagated through the connectivity of the adjacency matrix $\boldsymbol{W}$. We will discuss a relation between the adjacency matrix $\boldsymbol{W}$ and the underlying manifold in Sect. 4.2.2.

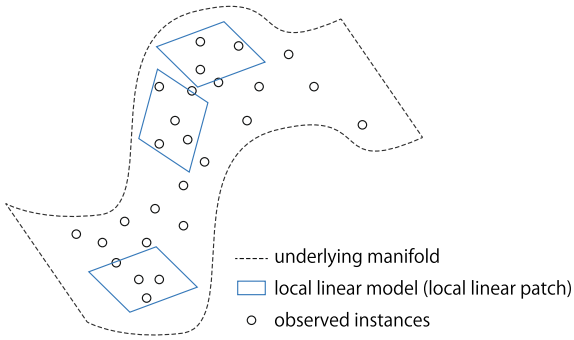


Fig. 1 An illustration of local linear approximation of a manifold. The underlying manifold, which can not be observed, is approximated by a set of local linear models (also called local linear patches)

3.2 Optimization

To optimize the problem (6), we can use any gradient-based algorithm (such as steepest descent and conjugate gradient) using the following gradient of the objective function with respect to σ_d:

$$\sum_{i=1}^{n} \frac{1}{D_{ii}} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^\top \left(\sum_j \frac{\partial W_{ij}}{\partial \sigma_d} \boldsymbol{x}_j - \frac{\partial D_{ii}}{\partial \sigma_d} \hat{\boldsymbol{x}}_i \right),$$

where

$$\hat{\boldsymbol{x}}_i = \frac{1}{D_{ii}} \sum_j W_{ij} \boldsymbol{x}_j.$$

The derivatives of W_{ij} and D_{ii} are

$$\frac{\partial W_{ij}}{\partial \sigma_d} = 2 W_{ij} (x_{id} - x_{jd})^2 \sigma_d^{-3},$$
$$\frac{\partial D_{ii}}{\partial \sigma_d} = \sum_j \frac{\partial W_{ij}}{\partial \sigma_d}.$$

Due to the non-convexity of the objective function, we cannot guarantee that solutions converge to the global optimal which means that the solutions depend on the initial σ_d. In our experiments, we employed well-known median heuristics (e.g., Gretton et al. 2007) for setting initial values of σ_d (Sect. 6). Another possible strategy is to use a number of different initial values for σ_d, which needs a high computational cost.

The gradient can be computed efficiently, due to the sparsity of the adjacency matrix. Since the number of edges of a k-NN graph is $O(nk)$, the derivative of adjacency matrix $\boldsymbol{W}$ can be calculated by $O(nkp)$. Then the entire derivative of the objective function can be calculated by $O(nkp^2)$. Note that k often takes a small value such as $k = 10$.

3.3 Normalization

The standard similarity function (4) cannot adapt to differences of the local scaling around each data point. These differences may cause a highly imbalanced adjacency matrix, in which W_{ij} has larger values around high density regions in the input space while W_{ij} has much smaller values around low density regions. As a result, labels in the high density regions will be dominantly propagated.

To overcome this problem we use symmetric normalized graph Laplacian and local scaling kernel. Symmetric normalized graph Laplacian (hereinafter, *normalized Laplacian* for short) is defined as

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}}.$$

The off-diagonal elements of this matrix is $-W_{ij}/\sqrt{D_{ii} D_{jj}}$, in which each similarity value is divided by the degrees of the connected two nodes. Another approach is to use local scaling kernel (Zelnik-Manor and Perona 2004) defined as (5), which can also be seen as a symmetric normalization. Let $\boldsymbol{S}$ be a diagonal matrix in which diagonal entries are defined as $S_{ii} = s_i^2$ (s_i is a local scaling parameter in (5)), and $\boldsymbol{\Delta}$ be a scaled distance matrix for $\boldsymbol{x}$ which means the elements of $\boldsymbol{\Delta}$ are $\sum_d (x_{id} - x_{jd})^2/\sigma_d$ if (i, j) has edges, and 0 otherwise. Then, local scaling kernel can be represented as

$$\boldsymbol{W} = \exp\left(-\boldsymbol{S}^{-\frac{1}{2}} \boldsymbol{\Delta} \boldsymbol{S}^{-\frac{1}{2}}\right)$$

where exp is applied to each element of matrix (note that it does not mean the matrix exponential, here). This means local scaling kernel normalizes the distances in the exponential of Gaussian kernel by a similar manner to symmetric normalized Laplacian. Instead of using the distance to the Kth neighbor to define s_i, the average distance of nearest-neighbors is also used (Jegou et al. 2007).

4 Analytical considerations

In Sect. 3, we defined our approach as the minimization of the local reconstruction error of input features. We here describe several interesting properties and interpretations of this definition.

4.1 Interpretation as feature propagation

First, we show that our objective function can be interpreted as a cross-validation error of the HGF model for the feature vector $\boldsymbol{x}$ on the graph. Let us divide a set of node indices $\{1, \ldots, n\}$ into a training set $\mathcal{T}$ and a validation set $\mathcal{V}$. Suppose that we try to predict $\boldsymbol{x}$ in the validation set $\{\boldsymbol{x}_i\}_{i \in \mathcal{V}}$ from the given training set $\{\boldsymbol{x}_i\}_{i \in \mathcal{T}}$ and the adjacency matrix $\boldsymbol{W}$. For this prediction problem, we consider the HGF model for $\boldsymbol{x}$:

$$\min_{\{\hat{\boldsymbol{x}}_i\}_{i=1}^n} \sum_{ij} W_{ij} \|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|_2^2. \tag{7}$$

$$\text{subject to } \hat{\boldsymbol{x}}_i = \boldsymbol{x}_i, \quad \text{for } i \in \mathcal{T},$$

where $\hat{\boldsymbol{x}}_i$ is a prediction for $\boldsymbol{x}_i$. Here, only $\hat{\boldsymbol{x}}_i$ in the validation set $\mathcal{V}$ is regarded as free variables in the optimization problem because the other $\{\hat{\boldsymbol{x}}_i\}_{i \in \mathcal{T}}$ is fixed at the observed values by the constraint. This can be interpreted as propagating $\{\boldsymbol{x}_i\}_{i \in \mathcal{T}}$ to predict $\{\boldsymbol{x}_i\}_{i \in \mathcal{V}}$. We call this process as *feature propagation*. The following matrix representation shows that the objective function of feature propagation has the same form as the basic objective function of the graph-based learning methods (2):

$$\min_{\hat{\boldsymbol{X}}} \text{trace}\left(\hat{\boldsymbol{X}}^\top \boldsymbol{L} \hat{\boldsymbol{X}}\right)$$

$$\text{subject to } \hat{x}_{ij} = x_{ij}, \text{ for } i \in \mathcal{T},$$

where $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n)^\top$, $\hat{\boldsymbol{X}} = (\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots \hat{\boldsymbol{x}}_n)^\top$, and x_{ij} and $\hat{x}_{ij}$ indicate (i, j)th entries of $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$ respectively.

When we employ leave-one-out as the cross-validation of the feature propagation model, we obtain

$$\sum_{i=1}^{n} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_{-i}\|_2^2, \tag{8}$$

where $\hat{\boldsymbol{x}}_{-i}$ is a prediction for $\boldsymbol{x}_i$ with $\mathcal{T} = \{1, \ldots, i-1, i+1, \ldots, n\}$ and $\mathcal{V} = \{i\}$. Due to the local averaging property of HGF (Zhu et al. 2003), we see $\hat{\boldsymbol{x}}_{-i} = \sum_j W_{ij}\boldsymbol{x}_j / D_{ii}$, and then (8) is equivalent to our objective function (6). From this equivalence, AEW can be interpreted as the optimization of parameters in graph weights of the HGF model for feature vectors through the leave-one-out cross-validation. This also means that our framework estimates labels using the adjacency matrix $\boldsymbol{W}$ optimized in the feature space instead of the output (label) space. Thus, if input features and labels share the same adjacency matrix (i.e., sharing the same local structure), the minimization of the objective function (6) should estimate the adjacency matrix by accurately propagating the labels of graph nodes.

4.2 Local linear approximation

The feature propagation model provides the interpretation of our approach as the optimization of the adjacency matrix under the assumption that $\boldsymbol{x}$ and y can be reconstructed by the same adjacency matrix. We here justify our approach in a more formal way from a viewpoint of local reconstruction with a lower dimensional manifold model.

4.2.1 Graph-based learning as local linear reconstruction

First, we show that all graph-based methods that we consider can be characterized by the local reconstruction for the outputs on the graph. The following proposition shows that all three methods which we reviewed in Sect. 2 have the local reconstruction property:

Proposition 1 *Assuming that we use unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, the optimal solutions of HGF, LLGC and spectral clustering can be represented as the following local averaging forms:*

– *HGF:*

$$F_{ik} = \frac{\sum_j W_{ij} F_{jk}}{D_{ii}} \quad \textit{for} \quad i = \ell + 1, \ldots, n.$$

– *LLGC:*

$$F_{ik} = \frac{\sum_j W_{ij} F_{jk}}{D_{ii} + \lambda} + \frac{\lambda Y_{ik}}{D_{ii} + \lambda} \quad \textit{for} \quad i = 1, \ldots, n.$$

– *Spectral clustering:*

$$F_{ik} = \frac{\sum_j W_{ij} F_{jk}}{D_{ii} - \rho_k} \quad \textit{for} \quad i = 1, \ldots, n,$$

where ρ_k is the kth smallest eigenvalue of $\boldsymbol{L}$.

Proof For HGF, the same equation was shown in Zhu et al. (2003). We here derive the reconstruction equations for LLGC and spectral clustering.

For LLGC, setting the derivatives to zero, we obtain

$$(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{F} + \lambda(\boldsymbol{F} - \boldsymbol{Y}) = \boldsymbol{0}.$$

From this equation, the reconstruction form of LLGC is obtained.

As is well known, the score $\boldsymbol{F}$ of spectral clustering can be calculated as the eigenvectors of graph Laplacian $\boldsymbol{L}$ (von Luxburg 2007) which are corresponding to the smallest c eigenvalues. We thereby obtain the following equation:

$$\boldsymbol{L}\boldsymbol{F} = \boldsymbol{F}\mathbf{P},$$

where $\mathbf{P} \in \mathbb{R}^{c\times c}$ is a diagonal matrix with the diagonal entry $\mathrm{P}_{ii} = \rho_i$. Substituting $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ to the above equation, we obtain

$$\boldsymbol{D}\boldsymbol{F} - \boldsymbol{F}\mathbf{P} = \boldsymbol{W}\boldsymbol{F}.$$

Since $\boldsymbol{D}$ and $\mathbf{P}$ are diagonal matrices, this equation derives the reconstruction form of spectral clustering. □

Regarding the optimization problems of the three methods as the minimization of the same score penalty term trace($\boldsymbol{F}\boldsymbol{L}\boldsymbol{F}$) under the different regularization strategies, which prevent a trivial solution $\boldsymbol{F} = \boldsymbol{0}$, it is reasonable that the similar reconstruction form is shared by those three methods. Among the three methods, HGF has the most standard form of local averaging. The output of the ith node is the weighted average over their neighbors connected by the graph edges. LLGC can be interpreted as a regularized variant of the local averaging. The averaging score $\boldsymbol{W}\boldsymbol{F}$ is regularized by the initial labeling $\boldsymbol{Y}$, and the balance of regularization is controlled by the parameter λ. Spectral clustering also has a similar form to the local reconstruction. The only difference here is that the denominator is modified by the eigenvalue of graph Laplacian. The eigenvalue ρ_k of graph Laplacian has a smaller value when the score matrix $\boldsymbol{F}$ has a smaller amount of variation on neighboring nodes. Spectral clustering thus has the same local reconstruction form in particular when the optimal scores have close values for neighboring nodes.

4.2.2 Error analysis

Proposition 1 shows that the graph-based learning algorithms can be regarded as local reconstruction methods. We next show the relationship between the local reconstruction error in the feature space described by our objective function (6) and the output space. For simplicity we consider the vector form of the score function $\boldsymbol{f} \in \mathbb{R}^n$ which can be considered as a special case of the score matrix $\boldsymbol{F}$, and discussions here can be applied to $\boldsymbol{F}$.

We assume the following manifold model for the input feature space, in which $\boldsymbol{x}$ is generated from corresponding some lower dimensional variable $\boldsymbol{\tau} \in \mathbb{R}^q$:

$$\boldsymbol{x} = g(\boldsymbol{\tau}) + \varepsilon_x,$$

where $g : \mathbb{R}^q \to \mathbb{R}^p$ is a smooth function, and $\varepsilon_x \in \mathbb{R}^p$ represents noise. In this model, y is also represented by some function form of $\boldsymbol{\tau}$:

$$y = h(\boldsymbol{\tau}) + \varepsilon_y,$$

where $h : \mathbb{R}^q \to \mathbb{R}$ is a smooth function, and $\varepsilon_y \in \mathbb{R}$ represents noise. For simplicity, we here consider a continuous output $y \in \mathbb{R}$ rather than discrete labels, and thus the latent function $h(\boldsymbol{\tau})$ is also defined as a continuous function. Our subsequent analysis is applicable to the

discrete label case by introducing an additional intermediate score variable which we will see in the end of this section. For this model, the following theorem shows the relationship between the reconstruction error of the feature vector $\boldsymbol{x}$ and the output y:

Theorem 1 *Suppose $\boldsymbol{x}_i$ can be approximated by its neighbors as follows*

$$\boldsymbol{x}_i = \frac{1}{D_{ii}} \sum_{j \sim i} W_{ij} \boldsymbol{x}_j + \boldsymbol{e}_i, \tag{9}$$

where $\boldsymbol{e}_i \in \mathbb{R}^p$ represents an approximation error. Then, the same adjacency matrix reconstructs the output $y_i \in \mathbb{R}$ with the following error:

$$y_i - \frac{1}{D_{ii}} \sum_{j \sim i} W_{ij} y_j = \boldsymbol{J} \boldsymbol{e}_i + O(\delta \boldsymbol{\tau}_i) + O(\varepsilon_x + \varepsilon_y), \tag{10}$$

where

$$\boldsymbol{J} = \frac{\partial h(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \left(\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \right)^+$$

with superscript $^+$ indicates pseudoinverse, and $\delta \boldsymbol{\tau}_i = \max_j (\| \boldsymbol{\tau}_i - \boldsymbol{\tau}_j \|_2^2)$.

Proof Let $\beta_j = W_{ij}/D_{ii}$ (Note that then $\sum_{j \sim i} \beta_j = 1$). Assuming that g is smooth enough, we obtain the following first-order Taylor expansion at $\boldsymbol{\tau}_i$ for the right hand side of (9).

$$\begin{aligned} \boldsymbol{x}_i = & \sum_{j \sim i} \beta_j \left(g(\boldsymbol{\tau}_i) + \frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) + O(\| \boldsymbol{\tau}_j - \boldsymbol{\tau}_i \|_2^2) \right) \\ & + \boldsymbol{e}_i + O(\varepsilon_x), \end{aligned}$$

Arranging this equation, we obtain

$$\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \sum_{j \sim i} \beta_j (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) = -\boldsymbol{e}_i + O(\delta \boldsymbol{\tau}_i) + O(\varepsilon_x).$$

If the Jacobian matrix $\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top}$ has full column rank, we obtain

$$\sum_{j \sim i} \beta_j (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) = -\left(\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \right)^+ \boldsymbol{e}_i + O(\delta \boldsymbol{\tau}_i) + O(\varepsilon_x). \tag{11}$$

On the other hand, we can see

$$\begin{aligned} \sum_{j \sim i} \beta_j y_j = & \sum_{j \sim i} \beta_j \left(h(\boldsymbol{\tau}_i) + \frac{\partial h(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) + O(\| \boldsymbol{\tau}_j - \boldsymbol{\tau}_i \|_2^2) \right) \\ & + O(\varepsilon_y) \\ = & y_i + \frac{\partial h(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \sum_{j \sim i} \beta_j (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) + O(\delta \boldsymbol{\tau}_i) + O(\varepsilon_y) \end{aligned} \tag{12}$$

Substituting (11) into (12), we obtain

$$\begin{aligned} y_i - \sum_{j \sim i} \beta_j y_j = & \frac{\partial h(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \left(\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \right)^+ \boldsymbol{e}_i \\ & + O(\delta \boldsymbol{\tau}_i) + O(\varepsilon_x + \varepsilon_y). \end{aligned}$$

□

From (10), we can see that the reconstruction error of y_i consists of three terms. The first term includes the reconstruction error for $\boldsymbol{x}_i$ which is represented by $\boldsymbol{e}_i$, and the second term is the distance between $\boldsymbol{\tau}_i$ and $\{\boldsymbol{\tau}_j\}_{j\sim i}$. Minimizing our objective function corresponds to reducing this first term, which means that reconstruction weights estimated in the input space provide an approximation of reconstruction weights in the output space. The two terms in (10) have a kind of trade-off relationship because we can reduce $\boldsymbol{e}_i$ if we use a lot of data points $\boldsymbol{x}_j$, but then $\delta\boldsymbol{\tau}_i$ would increase. The third term is the intrinsic noise which we cannot directly control.

A simple approach to exploit this theorem would be the regularization formulation, which can be a minimization of a combination of the reconstruction error for $\boldsymbol{x}$ and a penalization term for distances between data points connected by the edges. Regularized LLE (Wang et al. 2008; Cheng et al. 2009; Elhamifar and Vidal 2011; Kong et al. 2012) can be interpreted as one realization of such an approach. However, in the context of semi-supervised learning and un-supervised learning, selecting appropriate values of the regularization parameter for such a regularization term is difficult. We therefore optimize edge weights through the parameter of a similarity function, especially the bandwidth parameter of Gaussian similarity function σ. In this approach, a very large bandwidth (giving large weights to distant data points) may cause a large reconstruction error, while an extremely small bandwidth causes the problem of not giving enough weights to reconstruct.

For symmetric normalized graph Laplacian, we can not apply Theorem 1 to our algorithm. For example, in HGF, the local averaging relation for normalized Laplacian is $f_i = \sum_{j\sim i} W_{ij} f_j/\sqrt{D_{ii}D_{jj}}$. The following theorem is the normalized counterpart of Theorem 1:

Theorem 2 *Suppose that $\boldsymbol{x}_i$ can be approximated by its neighbors as follows*

$$\boldsymbol{x}_i = \sum_{j\sim i} \frac{W_{ij}}{\sqrt{D_{ii}D_{jj}}} \boldsymbol{x}_j + \boldsymbol{e}_i, \tag{13}$$

where $\boldsymbol{e}_i \in \mathbb{R}^p$ represents an approximation error. Then, the same adjacency matrix reconstructs the output $y_i \in \mathbb{R}$ with the following error:

$$y_i - \sum_{j\sim i} \frac{W_{ij}}{\sqrt{D_{ii}D_{jj}}} y_j = \left(1 - \sum_{j\sim i} \gamma_j\right)(h(\boldsymbol{\tau}_i) + \boldsymbol{J}g(\boldsymbol{\tau}_i)) + \boldsymbol{J}\boldsymbol{e}_i + O(\delta\boldsymbol{\tau}_i) + O(\varepsilon_x + \varepsilon_y), \tag{14}$$

where

$$\gamma_j = \frac{W_{ij}}{\sqrt{D_{ii}D_{jj}}}.$$

Proof The proof is almost the same as Theorem 1. However, the sum of the coefficients γ_j (corresponding to β_j in Theorem 1) cannot be 1. Applying the same Taylor expansion to the right hand side of (13), we obtain

$$\frac{\partial g(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \sum_{j\sim i} \gamma_j(\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) = -\boldsymbol{e}_i + (1 - \sum_{j\sim i} \gamma_j) g(\boldsymbol{\tau}_i) + O(\delta\boldsymbol{\tau}_i) + O(\varepsilon_x).$$

On the other hand, applying Taylor expansion to $y_i - \sum_{j\sim i} \gamma_j y_j$, we obtain

$$y_i - \sum_{j\sim i} \gamma_j y_j = (1 - \sum_{j\sim i} \gamma_j) h(\boldsymbol{\tau}_i) - \frac{\partial h(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}^\top} \sum_{j\sim i} \gamma_j (\boldsymbol{\tau}_j - \boldsymbol{\tau}_i) + O(\delta\boldsymbol{\tau}_i) + O(\varepsilon_y)$$

Using the above two equations, we obtain (14). □

Although Theorem 2 has a similar form to Theorem 1, we prefer Theorem 1 to Theorem 2 by the following reasons:

- Since the sum of the reconstruction coefficients $W_{ij}/\sqrt{D_{ii}D_{jj}}$ is no longer 1, the interpretation of local linear patch cannot be applied to (13).
- The reconstruction error (objective function) led by Theorem 2 (i.e., $\boldsymbol{x}_i - \sum_{j\sim i} W_{ij}\boldsymbol{x}_j/\sqrt{D_{ii}D_{jj}}$) results in a more complicated optimization.
- The error Eq. (14) in Theorem 2 has the additional term $(1 - \sum_{j\sim i} \gamma_j)(h(\boldsymbol{\tau}_i) + \boldsymbol{J}g(\boldsymbol{\tau}_i))$ compared to Theorem 1.

However, symmetric normalized graph Laplacian has been often preferred to the unnormalized one in many papers due to its practical performance advantage. For example, in semi-supervised learning, the balancing degree of each node would prevent only a small fraction of nodes from a dominant effect for propagating labels. Another example is in the context of spectral clustering, for which better convergence conditions of symmetric normalized graph Laplacian compared to the unnormalized one were proved by von Luxburg et al. (2008). We thus use symmetric normalized graph Laplacian as well in our experiments, though we do not change the objective function for reconstruction (6). Note that the normalization using local scaling kernel does not affect Theorem 1.

We have considered continuous outputs for simplicity, while our work can be extended to discrete outputs by redefining our output model, into which an intermediate label score variable f is incorporated:

$$f = h(\boldsymbol{\tau}) + \varepsilon_y.$$

The discrete label can be determined through f, following regular machine learning classification methods (for example, in binary classification: $y = 1$ if $f > 0.5$, and $y = 0$ if $f < 0.5$). According to the manifold assumption on this model, the label score f should have similar values for close data points on the manifold because the labels do not change drastically in the local region.

5 Related topics

We here describe relations of our approach with other related topics.

5.1 Relation to LLE

The objective function (6) is similar to the local reconstruction error of LLE (Roweis and Saul 2000), in which $\boldsymbol{W}$ is directly optimized as a real valued matrix. This manner has been used in many methods for graph-based semi-supervised learning and clustering (Wang and Zhang 2008; Daitch et al. 2009; Cheng et al. 2009; Liu et al. 2010), but LLE is very noise-sensitive (Chen and Liu 2011) and the resulting weights W_{ij} cannot necessarily represent

the similarity between the corresponding nodes (i, j). For example, for two nearly identical points $\boldsymbol{x}_{j_1}$ and $\boldsymbol{x}_{j_2}$, both connecting to $\boldsymbol{x}_i$, it is not guaranteed that W_{ij_1} and W_{ij_2} have similar values. To solve this problem, a regularization term can be introduced (Saul and Roweis 2003), while it is not easy to optimize the regularization parameter for this term. On the other hand, we optimize parameters of the similarity (kernel) function. This parameterized form of edge weights can alleviate the over-fitting problem. Moreover, obviously, the optimized weights still represent the node similarity.

5.2 Other hyper-parameter optimization strategies

AEW optimizes the parameters of graph edge weights without labeled instances. This property is powerful, especially for the case with only few (or no) labeled instances. Although several methods have been proposed for optimizing graph edge weights with standard model selection approaches (such as cross-validation and marginal likelihood maximization) by regarding them as usual hyper-parameters in supervised learning (Zhu et al. 2005; Kapoor et al. 2006; Zhang and Lee 2007; Muandet et al. 2009), most of those methods need labeled instances and become unreliable under the cases with few labels. Another approach is optimizing some criterion designed specifically for each graph-based algorithm (e.g., Ng et al. 2001; Zhu et al. 2003; Bach and Jordan 2004). Some of these criteria however have degenerate (trivial) solutions for which heuristics are proposed to prevent such solutions but the validity of those heuristics is not clear. Compared to these approaches, our approach is more general and flexible for problem settings, because AEW is independent of the number of classes (clusters), the number of labels, and the subsequent learning algorithms (the third step). In addition, model selection based approaches are basically for the third step in the three-step procedure, by which AEW can be combined with such methods, like that the optimized graph by AEW can be used as the input graph of these methods.

5.3 Graph construction

Besides k-NN, there have been several methods generating a graph (edges) from the feature vectors (e.g., Talukdar 2009; Jebara et al. 2009; Liu et al. 2010). Our approach can also be applied to those graphs because AEW only optimizes weights of edges. In our experiments, we used the edges of the k-NN graph as the initial graph of AEW. We then observed that AEW is not sensitive to the choice of k, comparing with usual k-NN graphs. This is because the Gaussian similarity value becomes small if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not close to each other to minimize the reconstruction error (6). In other words, redundant weights can be reduced drastically, because in the Gaussian kernel, weights decay exponentially according to the squared distance.

In the context of spectral clustering, a connection to statistical properties of graph construction has been analyzed (Maier et al. 2009, 2013). For example, Maier et al. (2013) have shown conditions under which the optimal convergence rate is achieved for a few types of graphs. These different studies also indicate that the quality of the input graph affects the final performance of learning methods.

5.4 Discussion on manifold assumption

A key assumption for our approach is *manifold assumption* which has been widely accepted in semi-supervised learning (e.g., see Chapelle et al. 2010). In manifold assumption, the input data is assumed to lie on a lower-dimensional manifold compared to the original

input space. Although verifying manifold assumption itself accurately would be difficult (because it is equivalent to estimating intrinsic dimensionality of the input data), the graph-based approach is known as a practical approximation of the underlying manifold which is applicable without knowing such a dimensionality. Many empirical evaluations have revealed that the manifold assumption-based approaches (most of them are graph-based) achieve high accuracy in various applications, particularly image and text data (see e.g., Patwari and Hero 2004; Lee and Kriegman 2005; Zhang and Zha 2005; Fergus et al. 2009; Chapelle et al. 2010; Aljabar et al. 2012). In these applications, the manifold assumption is reasonable, as implied by prior knowledge of the given data (e.g., in the face image classification, each person lies on different low-dimensional manifolds of pixels).

Another important implication in most of graph-based semi-supervised learning is *cluster assumption* (or *low density separation*) in which different classes are assumed to be separated by a low-density region. Graph-based approaches assume that nodes in the same class are densely connected while different classes are not so. If different classes are not separated by a low-density region, a nearest-neighbor graph would connect different classes which may cause miss-classification by propagating wrong class information. Several papers have addressed this problem (Wang et al. 2011; Gong et al. 2012). They considered the existence of singular points at which different classes of manifolds have intersections. Their approach is to measure similarity of two instances by considering similarity of tangent spaces of two instances, but this approach has to consider accurately modeling both of the local tangent and their similarity measure which introduces additional parameters and estimation errors. We perform experimental evaluation for this approach in Sect. 6.2.1.

6 Experiments

We evaluate the performance of our approach using synthetic and real-world datasets. AEW is applicable to all graph based learning methods reviewed in Sect. 2. We investigated the performance of AEW using the harmonic Gaussian field (HGF) model and local and global consistency (LLGC) model in semi-supervised learning, and using spectral clustering (SC) in un-supervised learning. For comparison in semi-supervised learning, we used *linear neighborhood propagation* (LNP) (Wang and Zhang 2008), which generates a graph using a LLE based objective function. LNP can have two regularization parameters, one of which is for the LLE process (the first and second steps in the three-step procedure), and the other is for the label estimation process (the third step in the three-step procedure). For the parameter in the LLE process, we used the heuristics suggested by Saul and Roweis (2003), and for the label propagation process, we chose the best parameter value in terms of the test accuracy. LLGC also has the regularization parameter in the propagation process (3), and we chose the best one again. This choice was to remove the effect by model selection and to compare the quality of the graphs directly. HGF does not have such hyper-parameters. All experimental results were averaged over 30 runs with randomly sampled data points.

6.1 Synthetic datasets

Using simple synthetic datasets in Fig. 2, we here illustrate the advantage of AEW by comparing the prediction performance in the semi-supervised learning scenario. Two datasets in Fig. 2 have the same form, but Fig. 2b has several noisy data points which may become

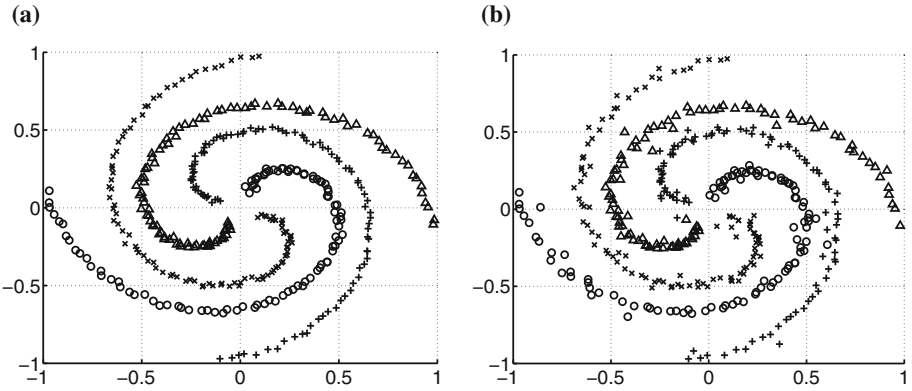


Fig. 2 Synthetic datasets

Table 1 Test error comparison for synthetic datasets

The best methods according to t-test with the significant level of 5% are highlighted with boldface

Dataset	k	HGF	AEW + HGF	LNP
(a)	10	0.057 (0.039)	**0.020 (0.027)**	0.039 (0.026)
(a)	20	0.261 (0.048)	**0.020 (0.028)**	0.103 (0.042)
(b)	10	0.119 (0.054)	**0.073 (0.035)**	0.103 (0.038)
(b)	20	0.280 (0.051)	**0.077 (0.035)**	0.148 (0.047)

bridge points (which can connect different classes, defined by Wang and Zhang 2008). In both cases, the number of classes is 4 and each class has 100 data points (thus, $n = 400$).

Table 1 shows the error rates for the unlabeled nodes of HGF and LNP under 0–1 loss. For HGF, we used the median heuristics to choose the parameter σ_d in similarity function (4), meaning that a common $\sigma (= \sigma_1 = \cdots = \sigma_p)$ is set as the median distance between all connected pairs of $\boldsymbol{x}_i$, and as the normalization of graph Laplacian, the symmetric normalization was used. The optimization of AEW started from the median σ_d. The results by AEW are shown in the column 'AEW + HGF' of Table 1. The number of labeled nodes was 10 in each class ($\ell = 40$, i.e., 10% of the entire datasets), and the number of neighbors in the graphs was set as $k = 10$ or 20.

In Table 1, we see HGF with AEW achieved better prediction accuracies than the median heuristics and LNP in all cases. Moreover, for both of datasets (a) and (b), AEW was most robust against the change of the number of neighbors k. This is because σ_d is automatically adjusted in such a way that the local reconstruction error is minimized and then weights for connections between different manifolds are reduced. Although LNP also minimizes the local reconstruction error, LNP may connect data points far from each other if it reduces the reconstruction error.

Figure 3 shows the graphs generated by (a) k-NN, (b) AEW, and (c) LNP, under $k = 20$ for the dataset of Fig. 2a. In this figure, the k-NN graph connects a lot of nodes in different classes, while AEW favorably eliminates those undesirable edges. LNP also has less edges between different classes compared to k-NN, but it still connects different classes. We can see that AEW shows the class structure more clearly, which can lead the better prediction performance of subsequent learning algorithms.

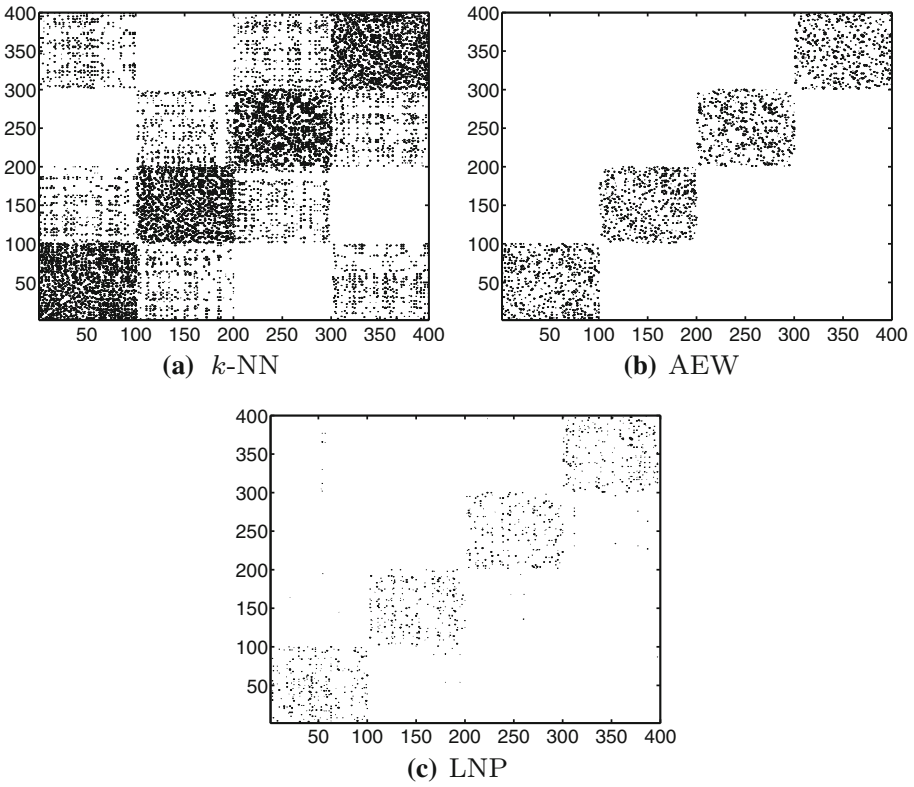


Fig. 3 Resultant graphs for the synthetic dataset of Fig. 2a ($k = 20$)

6.2 Real-world datasets

We here examine the performance of our proposed approach on the eight popular datasets shown in Table 2, namely COIL (COIL-20) (Nene et al. 1996), USPS (a preprocessed version from Hastie et al. 2001), MNIST (LeCun et al. 1998), ORL (Samaria and Harter 1994), Vowel (Asuncion and Newman 2007), Yale (Yale Face Database B) (Georghiades et al. 2001), optdigit (Asuncion and Newman 2007), and UMIST (Graham and Allinson 1998), for both semi-supervised learning and clustering tasks.

6.2.1 Semi-supervised learning

First, we show the results in the semi-supervised learning scenario using HGF. We evaluated three variants of the HGF model (three different normalizations), which are listed in the following:

- N-HGF. 'N' indicates normalized Laplacian, and the similarity function was (4) (not locally scaled).
- L-HGF. 'L' indicates local scaling kernel (5), and the graph Laplacian was not normalized.
- NL-HGF. 'NL' indicates that both of normalized Laplacian and local scaling kernel were used.

Table 2 List of datasets

	n	p	No. of classes
COIL	500	256	10
USPS	1000	256	10
MNIST	1000	784	10
ORL	360	644	40
Vowel	792	10	11
Yale	250	1200	5
Optdigit	1000	256	10
UMIST	518	644	20

For all three variants, the median heuristics was used to set σ_d (for local scaling kernel, we used median for scaled distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/s_i s_j$).

Figure 4 shows the test error for unlabeled nodes. In this figure, three dashed lines with different markers are by N-HGF, L-HGF and NL-HGF, while three solid lines with the same markers are by HGF with AEW. The performance difference within the variants of HGF was not large, compared to the effect of AEW, particularly in COIL, ORL, Vowel, Yale, and UMIST. We can rather see that AEW substantially improved the prediction accuracy of HGF in most cases. LNP is by the solid line without any markers. LNP outperformed HGF (without AEW, shown as the dashed lines) in COIL, ORL, Vowel, Yale and UMIST, while HGF with AEW (at least one of three variants) achieved better performance than LNP in all these datasets except for Yale (In Yale, LNP and HGF with AEW attained a similar accuracy). We further compared AEW with the following two baselines of hyper-parameter optimization approaches (both employing normalized Laplacian (NL) with the local scaling kernel):

1. Minimization of entropy (Zhu et al. 2003) with normalized Laplacian (NL-MinEnt): NL-MinEnt is represented by solid lines with the star shaped marker. We set the smoothing factor parameter ε in NL-MinEnt, which prevents the Gaussian width parameter from converging to 0, as $\varepsilon = 0.01$, and if the solution still converges to such a trivial solution, we increase ε by multiplying 10. The results of NL-MinEnt was unstable. Although we see that MinEnt stably improved the accuracy for all numbers of labels in the COIL dataset, NL-MinEnt sometimes largely deteriorated the accuracy, for example, in $\ell/\#\text{classes} = 1, 2, 3$, and 4 for the ORL dataset.
2. Cross-validation with normalized Laplacian (NL-CV): We employed 5-fold cross validation and a method proposed by Zhang and Lee (2007) which imposes an additional penalty to the CV error, defined as deviation of the Gaussian width parameter from pre-defined value $\tilde{\sigma}$. We used the median heuristic for $\tilde{\sigma}$, and selected an additional regularization parameter for hyper-parameter optimization from $\{0.001, 0.01, 0.1\}$ by 5-fold CV with different data partitioning (Note that we can not apply the CV approach for only one labeled instance for each class).

Compared to NL-HGF, Fig. 4 shows that any drastic change cannot be found for NL-CV under this small labeled instances setting. For example, in the Yale dataset, although CV improved the accuracy for all the numbers of labeled instances, the change was small compared to AEW.

Overall AEW-NL-HGF had the best prediction accuracy, where typical examples were USPS and MNIST. Although Theorem 1 exactly holds only for AEW-L-HGF among all

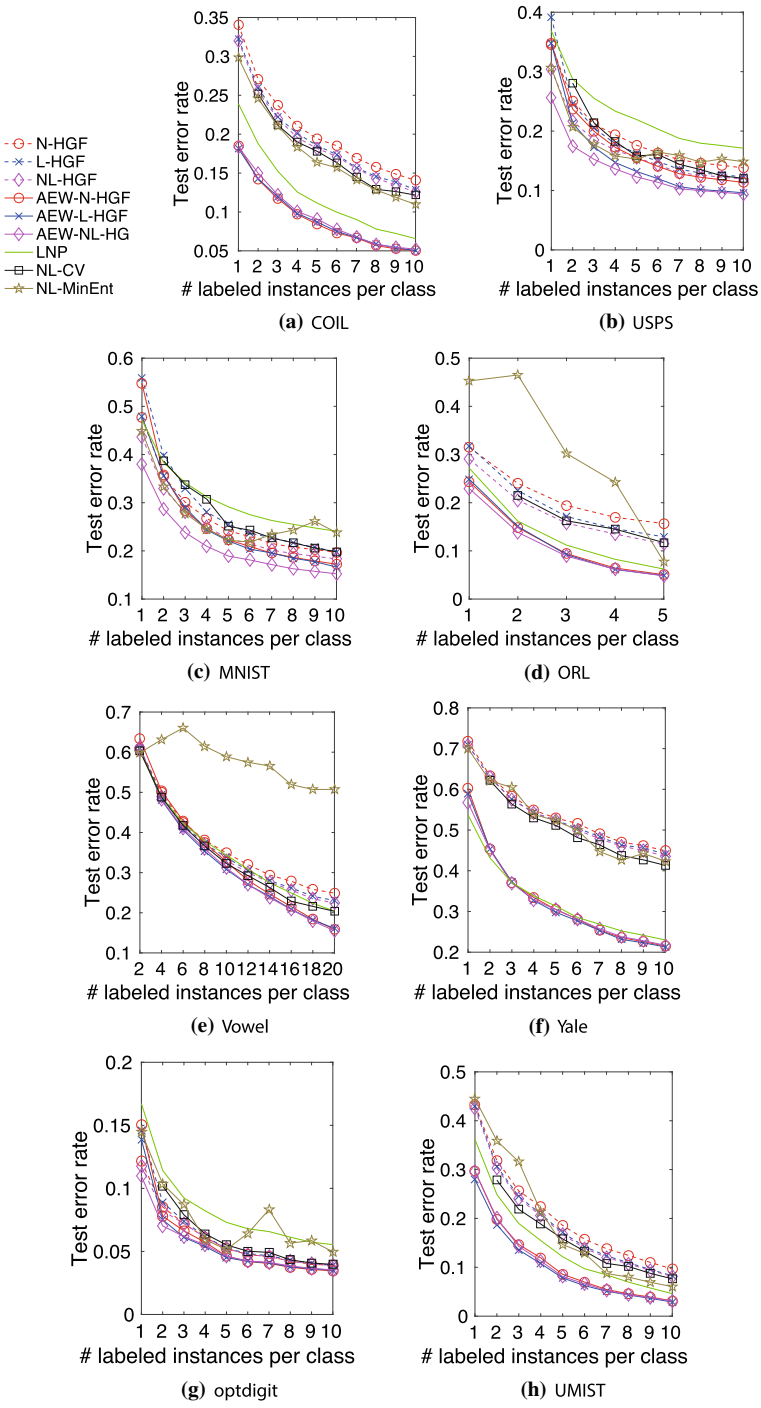


Fig. 4 Performance comparison using HGF on real-world datasets. For N-HGF, L-HGF, and NL-HGF, shown as the *dashed lines*, 'N' indicates normalized Laplacian, 'L' indicates local scaling kernel and 'NL' means that both normalized Laplacian and local scaling kernel are applied. HGFs with AEW are by *solid lines with markers*, while LNP is by a *solid line without any markers*

methods, we can see that AEW-NL-HGF, which is scaled for both Euclidian distance in the similarity and the degrees of the graph nodes, had more highly stable performance. This result suggests that balancing node weights (by normalized Laplacian) is practically advantageous to stabilize the propagation process of label information.

For further performance evaluation, we compared our approach with other state-of-the-art methods available for semi-supervised learning. We used the following two methods:

1. Spectral multi-manifold clustering (SMMC) (Wang et al. 2011): Wang et al. (2011) proposed to generate a similarity matrix using a similarity of *tangent spaces* for two instances. One of their main claims is that SMMC can handle intersections of manifolds which are difficult to deal with by using standard nearest-neighbor graph methods as we mentioned in Sect. 5.4. Although SMMC is proposed for a graph construction of spectral clustering, the resulting graph can be used for semi-supervised learning directly. We used the implementation by the authors,[2] and a recommended parameter setting in which the number of local probabilistic principal components analysis (PPCA) is $M = \lceil n/(10d) \rceil$, the number of neighbors is $k = 2\lceil \log(n) \rceil$, and a parameter of an exponential function in the similarity is $o = 8$, where d is a reduced dimension by PPCA which was set as $d = 10$ except for the Vowel dataset ($d = 5$ was used for the Vowel dataset because the original dimension is 10). Due to high computational complexity of SMMC which is at least $O(n^3 + n^2 dp^2)$ (see the paper for detail), we used those default settings instead of performing cross-validation.
2. Measure propagation (MP) (Subramanya and Bilmes 2011): MP estimates class assignment probabilities using a Kullback–Leibler (KL) divergence based criterion. We used the implementation provided by the authors.[3] Subramanya and Bilmes (2011) proposed an efficient alternating minimization procedure which enables us to perform cross-validation based model selection in the experiment. We tuned three parameters of MP including two regularization parameters μ and ν, and a kernel width parameter. The regularization parameters were selected from $\mu \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1\}$ and $\nu \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1\}$. For a kernel function, we employed the local scaling kernel, in which the width parameters were set as $\sigma_d = 10^a \sigma$, where σ is the value determined by the median heuristics and the parameter a were selected by cross-validation with 5 values uniformly taken from $[-1, 1]$. When we can not perform cross-validation because of the lack of labeled instances, we used $\mu = 10^{-2}$, $\nu = 10^{-2}$, and $a = 0$.

We here employed AEW-NL-HGF as a representative of our approach (the results of AEW-NL-HGF were the same as Fig. 4). Figure 5 shows the results on the comparison with the above two methods for semi-supervised learning approaches. We can see that AEW-NL-HGF had clearly better performance except for the optdigit dataset in which MP has similar prediction accuracy to AEW-NL-HGF. Also our approach outperformed SMMC, which considers possible intersections of manifolds.

Next, we used the LLGC model for comparison. Table 3 shows the test error rates for the eight datasets in Table 2. Here again, we can see that AEW improved the test error rates of LLGC, and here AEW-L-LLGC showed the best performance in all cases except only one case (MNIST ℓ/#classes $= 5$). In this table, the symbol '*' means that AEW improved the accuracy from the corresponding method without AEW (which is shown in one of the left three methods) in terms of t-test with the significance level of 5%. AEW improved the prediction performance of LLGC except Vowel under ℓ/#class $= 2$, and all three methods with AEW outperformed LNP in all 24 ($= 3 \times 8$) cases except only one exception.

[2] http://lamda.nju.edu.cn/code_SMMC.ashx.

[3] http://melodi.ee.washington.edu/mp/.

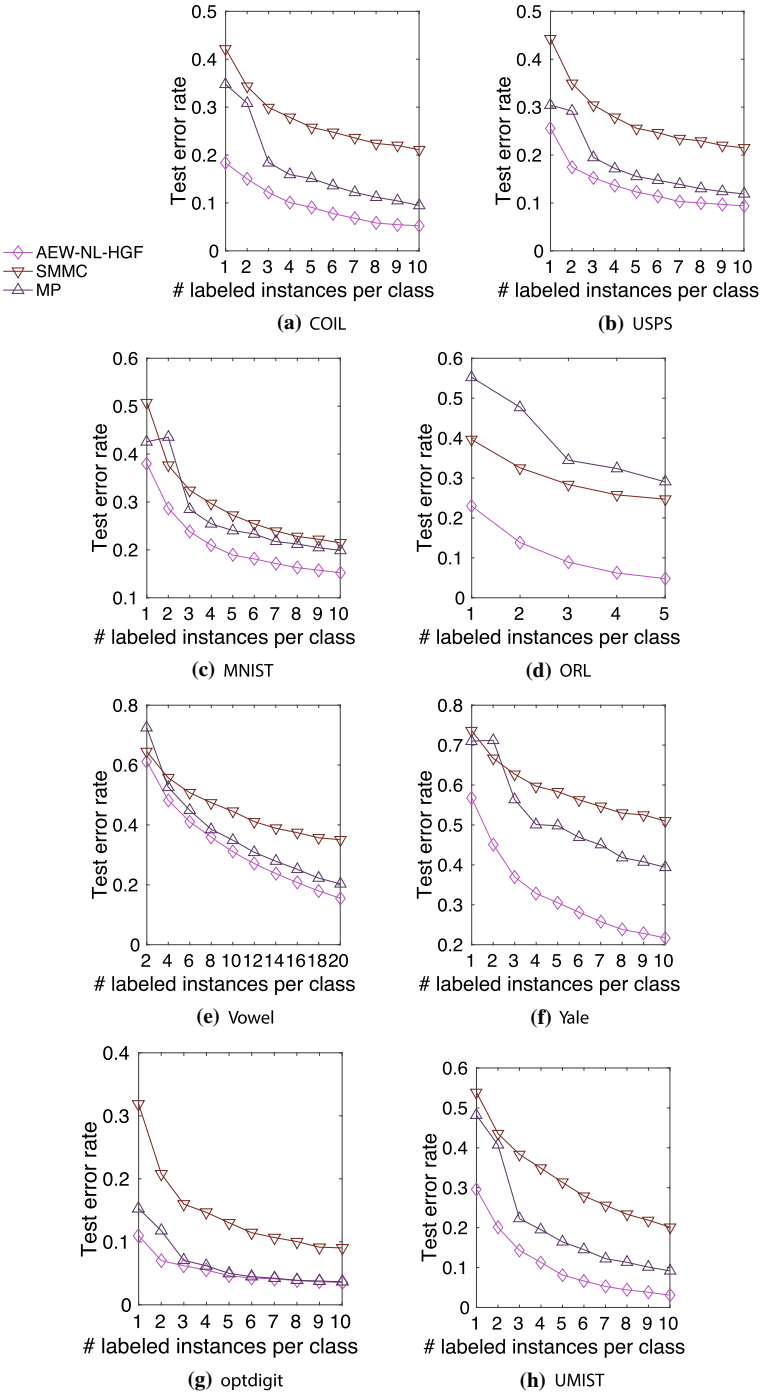


Fig. 5 Performance comparison with other semi-supervised methods

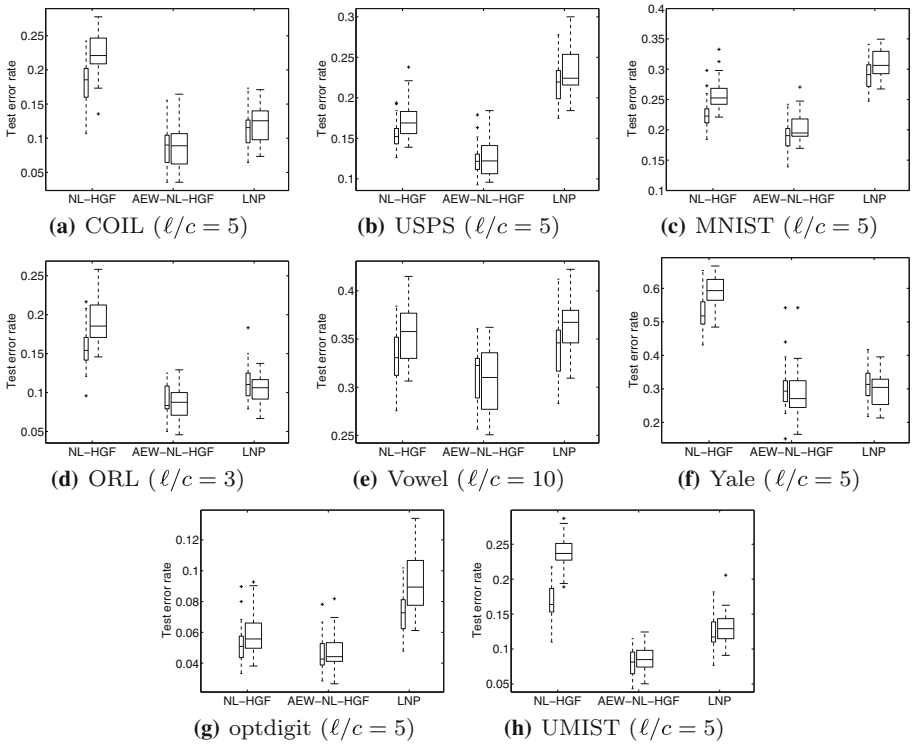
Table 3 Test error rate comparison using LLGC (averages on 30 runs and their standard deviation)

ℓ/#classes	N-LLGC	L-LLGC	NL-LLGC	AEW-N-LLGC	AEW-L-LLGC	AEW-NL-LLGC	LNP
COIL 1	0.310 (0.047)	0.296 (0.046)	0.296 (0.046)	0.176 (0.031)*	**0.158 (0.035)***	0.172 (0.035)*	0.239 (0.027)
5	0.179 (0.034)	0.170 (0.033)	0.170 (0.034)	**0.087 (0.029)***	**0.084 (0.033)***	**0.086 (0.032)***	0.111 (0.027)
10	0.116 (0.026)	0.112 (0.024)	0.110 (0.026)	**0.047 (0.021)***	**0.050 (0.020)***	**0.051 (0.018)***	0.066 (0.022)
USPS 1	0.290 (0.068)	0.258 (0.064)	0.271 (0.066)	0.263 (0.060)*	**0.220 (0.061)***	0.241 (0.063)*	0.369 (0.078)
5	0.174 (0.018)	0.152 (0.017)	0.155 (0.017)	0.150 (0.019)*	**0.121 (0.018)***	0.127 (0.018)*	0.219 (0.026)
10	0.140 (0.013)	0.123 (0.014)	0.124 (0.012)	0.117 (0.016)*	**0.096 (0.012)***	**0.097 (0.012)***	0.171 (0.018)
MNIST 1	0.420 (0.053)	0.424 (0.056)	0.404 (0.053)	0.387 (0.051)*	**0.356 (0.054)***	**0.361 (0.048)***	0.471 (0.038)
5	0.242 (0.019)	0.244 (0.026)	0.229 (0.019)	0.216 (0.020)*	0.212 (0.024)*	**0.194 (0.022)***	0.291 (0.025)
10	0.204 (0.017)	0.199 (0.017)	0.192 (0.018)	0.179 (0.017)*	**0.167 (0.016)***	**0.161 (0.016)***	0.241 (0.022)
ORL 1	0.282 (0.026)	0.262 (0.026)	0.265 (0.027)	0.230 (0.023)*	**0.212 (0.028)***	0.224 (0.028)*	0.272 (0.026)
3	0.171 (0.025)	0.142 (0.024)	0.137 (0.024)	0.092 (0.022)*	**0.084 (0.021)***	**0.086 (0.022)***	0.112 (0.023)
5	0.140 (0.020)	0.106 (0.018)	0.101 (0.019)	**0.052 (0.017)***	**0.049 (0.016)***	**0.049 (0.017)***	0.062 (0.016)
Vowel 2	0.584 (0.032)	**0.577 (0.029)**	0.581 (0.031)	0.583 (0.030)	**0.573 (0.030)**	**0.575 (0.028)***	0.601 (0.029)
10	0.337 (0.030)	0.326 (0.031)	0.325 (0.029)	0.314 (0.027)*	**0.303 (0.031)***	**0.306 (0.029)***	0.344 (0.032)
20	0.232 (0.023)	0.210 (0.022)	0.206 (0.023)	**0.161 (0.023)***	**0.160 (0.023)***	**0.159 (0.023)***	0.204 (0.024)
Yale 1	0.679 (0.052)	0.678 (0.056)	0.675 (0.060)	**0.554 (0.060)***	**0.539 (0.082)***	**0.544 (0.072)***	**0.537 (0.075)**
5	0.490 (0.043)	0.481 (0.046)	0.484 (0.045)	0.309 (0.057)*	**0.292 (0.061)***	0.297 (0.059)*	0.313 (0.048)
10	0.394 (0.041)	0.385 (0.044)	0.386 (0.041)	0.226 (0.038)*	**0.217 (0.040)***	0.224 (0.040)*	0.230 (0.037)
Optdigits 1	0.108 (0.040)	0.102 (0.038)	0.107 (0.039)	**0.093 (0.038)***	**0.087 (0.034)***	0.095 (0.034)*	0.167 (0.065)
5	0.051 (0.012)	0.051 (0.012)	0.050 (0.012)	**0.043 (0.011)***	**0.044 (0.012)***	**0.044 (0.011)***	0.073 (0.014)
10	0.041 (0.006)	0.039 (0.006)	0.040 (0.006)	**0.035 (0.007)***	**0.035 (0.007)***	**0.034 (0.006)***	0.055 (0.011)

Table 3 continued

ℓ/#classes	N-LLGC	L-LLGC	NL-LLGC	AEW-N-LLGC	AEW-L-LLGC	AEW-NL-LLGC	LNP
UMIST 1	0.423 (0.032)	0.406 (0.032)	0.411 (0.032)	0.292 (0.035)*	**0.255 (0.036)***	0.274 (0.042)*	0.362 (0.028)
5	0.172 (0.020)	0.165 (0.019)	0.165 (0.019)	0.099 (0.017)*	**0.089 (0.017)***	0.095 (0.017)*	0.122 (0.023)
10	0.090 (0.018)	0.080 (0.017)	0.081 (0.017)	0.040 (0.014)*	**0.035 (0.013)***	0.039 (0.013)*	0.046 (0.017)

The best method in each row is highlighted with boldface. Any other method, which is comparable with the best method in the same row, according to t-test with the significant level of 5%, is also highlighted by boldface. The symbol '*' means that AEW significantly improved the performance of the corresponding original LLGC (which is shown in the left three columns), according to t-test with the same significance level



(a) COIL ($\ell/c = 5$) **(b)** USPS ($\ell/c = 5$) **(c)** MNIST ($\ell/c = 5$)

(d) ORL ($\ell/c = 3$) **(e)** Vowel ($\ell/c = 10$) **(f)** Yale ($\ell/c = 5$)

(g) optdigit ($\ell/c = 5$) **(h)** UMIST ($\ell/c = 5$)

Fig. 6 Comparison in test error rates of $k = 20$ with $k = 10$. Two boxplots of each method correspond to $k = 10$ in the *left* (with a smaller width) and $k = 20$ in the *right* (with a larger width). The number of labeled instances in each class is represented as ℓ/c

In Table 3, AEW-L-LLGC or AEW-NL-LLGC (i.e., AEW with local scaling kernel) achieved the lowest test error rates in 22 out of all 24 cases. This result suggests that incorporating differences of local scaling is important for real-world datasets. We can also see that AEW-NL-LLGC, for which Theorem 1 does not hold exactly, shows the best in 14 cases (highlighted by boldface) and the second best in 9 cases out of all 24 cases.

We further examined the effect of k. Figure 6 shows the test error rates for $k = 20$ and 10, using NL-HGF, AEW-NL-HGF, and LNP. The number of labeled instances in each dataset is the midst value in the horizontal axis of Fig. 4. We can see that AEW was most robust among the three methods, meaning that the test error rate was not sensitive to k. In all cases, the median performance (the horizontal line in each box) of NL-HGF with $k = 20$ was worse than that with $k = 10$. On the other hand, AEW-NL-HGF with $k = 20$ had a similar (or clearly better) median to that with $k = 10$ in each case. The performance of LNP was also deteriorated substantially by letting $k = 20$ in some cases, such as Vowel and optdigit.

6.2.2 Clustering

We also demonstrate the effectiveness of AEW in an un-supervised learning scenario, especially in clustering. Table 4 shows *adjusted Rand index* (ARI) (Hubert and Arabie 1985) of each of nine compared method. ARI was extended from Rand index (Rand 1971), which

Table 4 Clustering performance comparison using ARI (averages on 30 runs and their standard deviation)

	N-SC	L-SC	NL-SC	AEW-N-SC	AEW-L-SC	AEW-NL-SC	kernel SMCE	SMMC	k-means	k-means
COIL	0.545	0.508	0.546	**0.717***	0.700*	**0.738***	0.488	0.429	0.453	0.445
	(0.040)	(0.040)	(0.043)	(0.047)	(0.058)	(0.055)	(0.044)	(0.027)	(0.037)	(0.041)
USPS	.585	0.523	0.600	0.619*	0.553	**0.640***	0.537	0.438	0.505	0.518
	(0.030)	(0.122)	(0.032)	(0.044)	(0.119)	(0.044)	(0.033)	(0.071)	(0.031)	(0.032)
MNIST	0.387	0.082	0.405	0.417*	**0.469***	0.444*	0.383	**0.423**	0.349	0.333
	(0.034)	(0.024)	(0.033)	(0.047)	(0.047)	(0.051)	(0.031)	(0.122)	(0.031)	(0.033)
ORL	0.641	0.539	**0.666**	0.624	0.533	**0.654**	0.577	0.564	0.614	0.503
	(0.021)	(0.115)	(0.024)	(0.025)	(0.094)	(0.031)	(0.028)	(0.018)	(0.030)	(0.037)
Vowel	0.185	0.168	0.187	0.131	0.146	0.193	0.160	0.157	**0.212**	**0.214**
	(0.017)	(0.023)	(0.018)	(0.051)	(0.033)	(0.016)	(0.012)	(0.042)	(0.009)	(0.015)
Yale	0.092	0.087	0.099	**0.302***	0.171*	0.280*	**0.291**	0.073	0.002	0.005
	(0.016)	(0.014)	(0.016)	(0.048)	(0.052)	(0.053)	(0.059)	(0.013)	(0.007)	(0.009)
Optdigits	**0.873**	0.810	**0.887**	**0.889**	0.821	**0.881**	0.799	0.423	0.674	0.700
	(0.039)	(0.116)	(0.040)	(0.042)	(0.083)	(0.034)	(0.030)	(0.122)	(0.022)	(0.009)
UMIST	0.426	0.385	0.425	**0.560***	0.450*	0.524*	0.245	0.350	0.336	0.338
	(0.024)	(0.036)	(0.023)	(0.052)	(0.103)	(0.047)	(0.025)	(0.014)	(0.020)	(0.019)

The best method in each row is highlighted with boldface. Any other method, which is comparable with the best method in the same row, according to t-test with the significance level of 5%, is also highlighted with boldface. The symbol '*' means that AEW significantly improved the performance of the corresponding SC (which is in the left three columns), according to the t-test with the same significance level

evaluates the accuracy of clustering results based on pairwise comparison, in such a way that the expected value for random labeling of ARI should take 0 (and the maximum is 1). We employed spectral clustering (SC) as a graph-based clustering to which AEW applies, and for comparison, we used *sparse manifold clustering and embedding* (SMCE) (Elhamifar and Vidal 2011), k-means clustering, and kernel k-means clustering. SMCE generates a graph using a LLE based objective function. The difference from LLE is that SMCE prevents connections between different manifolds by penalizing edge weights with the weighted L_1 norm, according to distances between node pairs. However, as often the case with LLE, there are no reliable ways to select the regularization parameter. For comparison, we further used SMMC again here because a graph created by this method is also applicable to spectral clustering, and the same parameter settings as the semi-supervised case were used.

In Table 4, we can see that either AEW-N-SC or AEW-NL-SC achieved the best ARI for all datasets except Vowel. These two methods (AEW-N-SC and AEW-NL-SC) significantly improved the performance of SC with k-NN graphs in four datasets, i.e. COIL, USPS, Yale, and UMIST. L-SC and AEW-L-SC, i.e. SC and SC with AEW, both using unnormalized graph Laplacian, were not comparable to those with normalized graph Laplacian. AEW-L-SC significantly improved the performance of L-SC in three datasets, while AEW-L-SC was not comparable to AEW-N-SC and AEW-NL-SC. These results suggest that the normalization of Laplacian is important for SC, being compared to label propagation. As a result, AEW-NL-SC showed the most stable performance among the three methods with AEW. In Vowel, k-means and kernel k-means achieved the best performance, suggesting that the lower-dimensional manifold model assumed in Theorem 1 is not suitable for this dataset. SMMC achieved comparable performance against the competing methods for MNIST only. The difficulty of this method would be the parameter tuning. Overall however we emphasize that spectral clustering with AEW achieved the best performance for all datasets except Vowel.

7 Conclusions

We have proposed the *adaptive edge weighting* (AEW) method for graph-based learning algorithms such as label propagation and spectral clustering. AEW is based on the minimization of the local reconstruction error under the constraint that each edge has the function form of similarity for each pair of nodes. Due to this constraint, AEW has numerous advantages against LLE based approaches, which have a similar objective function. For example, noise sensitivity of LLE can be alleviated by the parameterized form of the edge weights, and the similarity form for the edge weights is very reasonable for many graph-based methods. We also provide several interesting properties of AEW, by which our objective function can be motivated analytically. Experimental results have demonstrated that AEW can improve the performance of graph-based algorithms substantially, and we also saw that AEW outperformed LLE based approaches in almost all cases.

Acknowledgements M.K. has been partially supported by JSPS KAKENHI 26730120, and H.M. has been partially supported by MEXT KAKENHI 16H02868 and FiDiPro, Tekes.

References

Aljabar, P., Wolz, R., & Rueckert, D. (2012). Manifold learning for medical image registration, segmentation, and classification. In *Machine learning in computer-aided diagnosis: Medical imaging intelligence and analysis*. IGI Global.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bach, F. R., & Jordan, M. I. (2004). Learning spectral clustering. In S. Thrun, L. K. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems* (Vol. 16). Cambridge, MA: MIT Press.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bengio, Y., Delalleau, O., & Le Roux, N. (2006). Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, & A. Zien (Eds.), *Semi-supervised learning* (pp. 193–216). Cambridge, MA: MIT Press.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In C. E. Brodley & A. P. Danyluk (Eds.), *Proceedings of the 18th international conference on machine learning* (pp. 19–26). Los Altos, CA: Morgan Kaufmann.

Chapelle, O., Schlkopf, B., & Zien, A. (2010). *Semi-supervised learning* (1st ed.). Cambridge, MA: The MIT Press.

Chen, J., & Liu, Y. (2011). Locally linear embedding: A survey. *Artificial Intelligence Review*, *36*, 29–48.

Cheng, H., Liu, Z., & Yang, J. (2009). Sparsity induced similarity measure for label propagation. In *IEEE 12th international conference on computer vision* (pp. 317–324). Piscataway, NJ: IEEE.

Chung, F. R. K. (1997). *Spectral graph theory*. Providence, RI: American Mathematical Society.

Daitch, S. I., Kelner, J. A., & Spielman, D. A. (2009). Fitting a graph to vector data. In *Proceedings of the 26th international conference on machine learning* (pp. 201–208). New York, NY: ACM.

Elhamifar, E., & Vidal, R. (2011). Sparse manifold clustering and embedding. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 24, pp. 55–63).

Fergus, R., Weiss, Y., & Torralba, A. (2009). Semi-supervised learning in gigantic image collections. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 22, pp. 522–530). Red Hook, NY: Curran Associates Inc.

Georghiades, A., Belhumeur, P., & Kriegman, D. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 643–660.

Gong, D., Zhao, X., & Medioni, G. G. (2012). Robust multiple manifold structure learning. In *The 29th international conference on machine learning*. icml.cc/Omnipress.

Graham, D. B., & Allinson, N. M. (1998). Characterizing virtual eigensignatures for general purpose face recognition. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, & T. S. Huang (Eds.), *Face recognition: From theory to applications; NATO ASI Series F, computer and systems sciences* (Vol. 163, pp. 446–456).

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. J. (2007). A kernel method for the two-sample-problem. In B. Schölkopf, J. C. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems* (Vol. 19, pp. 513–520). Cambridge, MA: MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. New York, NY: Springer.

Herbster, M., Pontil, M., & Wainer, L. (2005). Online learning over graphs. In *Proceedings of the 22nd annual international conference on machine learning* (pp. 305–312). New York, NY: ACM.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193–218.

Jebara, T., Wang, J., & Chang, S.-F. (2009). Graph construction and b-matching for semi-supervised learning. In A. P. Danyluk, L. Bottou, & M. L. Littman (Eds.), *Proceedings of the 26th annual international conference on machine learning* (p. 56). New York, NY: ACM.

Jegou, H., Harzallah, H., & Schmid, C. (2007). A contextual dissimilarity measure for accurate and efficient image search. In *2007 IEEE computer society conference on computer vision and pattern recognition*. Washington, DC: IEEE Computer Society.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In T. Fawcett & N. Mishra (Eds.), *Machine learning, proceedings of the 20th international conference* (pp. 290–297). Menlo Park, CA: AAAI Press.

Kapoor, A., Qi, Y. A., Ahn, H., & Picard, R. (2006). Hyperparameter and kernel learning for graph based semi-supervised classification. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems* (Vol. 18, pp. 627–634). Cambridge, MA: MIT Press.

Karasuyama, M., & Mamitsuka, H. (2013). Manifold-based similarity adaptation for label propagation. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 26, pp. 1547–1555). Red Hook, NY: Curran Associates Inc.

Kong, D., Ding, C. H., Huang, H., & Nie, F. (2012). An iterative locally linear embedding algorithm. In J. Langford & J. Pineau (Eds.), *Proceedings of the 29th international conference on machine learning* (pp. 1647–1654). New York, NY: Omnipress.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, K.-C., & Kriegman, D. (2005). Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In *IEEE conference on computer vision and pattern recognition* (pp. 852–859).

Liu, W., He, J., & Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning* (pp. 679–686). New York, NY: Omnipress.

Maier, M., von Luxburg, U., & Hein, M. (2009). Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems* (Vol. 21, pp. 1025–1032). Red Hook, NY: Curran Associates, Inc.

Maier, M., von Luxburg, U., & Hein, M. (2013). How the result of graph clustering methods depends on the construction of the graph. *ESAIM: Probability and Statistics*, *17*, 370–418.

Meila, M., & Shi, J. (2001). A random walks view of spectral segmentation. In T. Jaakkola & T. Richardson (Eds.), *Proceedings of the eighth international workshop on artifical intelligence and statistics*. Los Altos, CA: Morgan Kaufmann.

Muandet, K., Marukatat, S., & Nattee, C. (2009). Robust graph hyperparameter learning for graph based semi-supervised classification. In *13th Pacific-Asia conference advances in knowledge discovery and data mining* (pp. 98–109).

Nene, S. A., Nayar, S. K., & Murase, H. (1996). Columbia object image library (COIL-20). Technical report, Technical Report CUCS-005-96.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems* (Vol. 14, pp. 849–856). Cambridge, MA: MIT Press.

Patwari, N., & Hero, A. O. (2004). Manifold learning algorithms for localization in wireless sensor networks. In *IEEE international conference on acoustics, speech, and signal processing* (Vol. 3, pp. iii–857–60).

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846–850.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Samaria, F., & Harter, A. (1994). Parameterisation of a stochastic model for human face identification. In *Proceedings of the second IEEE workshop on applications of computer vision* (pp. 138–142).

Saul, L. K., & Roweis, S. T. (2003). Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, *4*, 119–155.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: From transductive to semi-supervised learning. In L. D. Raedt & S. Wrobel (Eds.), *Proceedings of the 22nd international conference on machine learning* (pp. 824–831). New York, NY: ACM.

Spielman, D. A., & Teng, S.-H. (2004). Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In L. Babai (Ed.), *Proceedings of the 36th annual ACM symposium on theory of computing*. New York, NY: ACM.

Subramanya, A., & Bilmes, J. A. (2011). Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, *12*, 3311–3370.

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with Markov random walks. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems* (Vol. 14, pp. 945–952). Cambridge, MA: MIT Press.

Talukdar, P. P. (2009). Topics in graph construction for semi-supervised learning. Technical report MS-CIS-09-13, University of Pennsylvania, Department of Computer and Information Science.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

von Luxburg, U., Belkin, M., & Bousquet, O. (2008). Consistency of spectral clustering. *Annals of Statistics*, *36*(2), 555–586.

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

Wang, G., Yeung, D.-Y., & Lochovsky, F. H. (2008). A new solution path algorithm in support vector regression. *IEEE Transactions on Neural Networks*, *19*(10), 1753–1767.

Wang, Y., Jiang, Y., Wu, Y., & Zhou, Z. H. (2011). Spectral clustering on multiple manifolds. *IEEE Transactions on Neural Networks*, *22*(7), 1149–1161.

Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. In *Advances in neural information processing systems* (Vol. 17, pp. 1601–1608). Cambridge, MA: MIT Press.

Zhang, X., & Lee, W. S. (2007). Hyperparameter learning for graph based semi-supervised learning algorithms. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems* (Vol. 19, pp. 1585–1592). Cambridge, MA: MIT Press.

Zhang, Z., & Zha, H. (2005). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, *26*(1), 313–338.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems* (Vol. 16). Cambridge, MA: MIT Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In T. Fawcett & N. Mishra (Eds.), *Proceedings of the twentieth international conference on machine learning* (pp. 912–919). Menlo Park, CA: AAAI Press.

Zhu, X., Kandola, J., Ghahramani, Z., & Lafferty, J. (2005). Nonparametric transforms of graph kernels for semi-supervised learning. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 17, pp. 1641–1648). Cambridge, MA: MIT Press.