Int. J. Advance Soft Compu. Appl, Vol. 10, No. 3, November 2018
ISSN 2074-8523

WEIGHTED UN-NORMALIZED HYPERGRAPH LAPLACIAN EIGENMAPS FOR CLASSIFICATION PROBLEMS

Loc Tran, An Mai, Tho Quan amd Linh Tran

John von Neumann Institute, VNU-HCM
e-mail: tran0398@umn.edu
e-mail: an.mai@jvn.edu.vn
Ho Chi Minh City University of Technology, VNU-HCM
e-mail: qttho@hcmut.edu.vn
Thu Dau Mot University
e-mail: linhtran.cntt@tdmu.edu.vn

Abstract

In the past, many dimensional reduction methods such as Local Linear Embedding (LLE) and Laplacian Eigenmaps (LE) have been successfully developed. However, in many real world applications, representing the dataset as un-directed graph, used in Laplacian Eigenmaps and Local Linear Embedding methods, is not complete. Approximating complex relationship as pairwise will lead to the loss of information. The natural way overcoming the information loss is to represent the dataset as the hypergraph. However, representing the dataset as the hypergraph will not lead to the perfection. The number of hyper-edges may be large; hence this will lead to high time complexity of the clustering methods or the classification methods when we try to apply the clustering/classification methods to this hypergraph dataset. Thus, in this paper, we develop the un-normalized hypergraph Laplacian Eigenmaps. Moreover, in the developed un-normalized hypergraph Laplacian Eigenmaps algorithm, we assume that the weights of all hyper-edges are equal to 1. This is not true at all in practical applications. Thus, in this paper, we will also develop the weighted un-normalized hypergraph Laplacian Eigenmaps and the weighted hypergraph semi-supervised learning method. Experimental results show that the weighted hyper-graph semi-supervised learning method achieves the highest accuracy performance measures.

Keywords: *hypergraph, Laplacian, Eigenmaps, weighted, semi-supervised, learning.*

1 Introduction

Recently, our capability to collect and store data has far exceeded our capability to analyze it. In problems such as face recognition and biological network inference problem using gene expression data, we are given a large dataset in which each observation contains a large number of variables. This number of variables is called the dimension of each observation. In this given setting, the data lies in a high-dimensional space. For example, the $256 * 256$ image has 65536 dimensions if we treat each pixel as one variable. It's hard for human beings to visualize and understand these high dimensional data because of limited computing resources. Moreover, it turns out that not all variables are needed for understanding the primary phenomenon. In the other words, there is a high degree of redundancy in the data they represent. Hence, the structure and the content of the data may be captured by a lesser set of variables. There are also may be too much noise in the data. Hence there is a need to reduce the dimensionality of the data (i.e. reduce the noise of the data) before we apply the clustering (i.e. un-supervised learning) methods and classification (semi-supervised learning and supervised learning) methods to the dataset. In the other words, we can build the more effective data analysis tools. Those are why we need to develop the dimensional reduction methods.

In our literature review, many dimensional reduction methods have been successfully developed and applied to various applications such as speech recognition, face recognition, and biological network inference problem using gene expression data, to name a few. There are two classes of dimensional reduction methods which are the linear and the non-linear techniques [1]. Linear dimensional reduction methods assume that the data lies on or close to linear subspace of the high-dimensional ambient space. Linear dimensional reduction methods have been developed and used for a long time. For example, Principle Component Analysis (i.e. PCA) was developed in 1901 and is still the most widely used dimensional reduction methods nowadays. For instance, the PCA

technique is employed in and successfully applied to speech recognition research field [2], face recognition research field [3], and biological network inference research field [4]. In the other hand, non-linear dimensional reduction methods make no assumption about the linearity and are designed to recognize complex non-linear manifolds as well as linear ones. Recently, many researchers have focused on developing various non-linear dimensional reduction methods such as Kernel PCA [5], Isomap [6], Local Linear Embedding [7], and Laplacian Eigenmaps [8].

However, in many real world applications, representing the dataset as un-directed graph, used in Laplacian Eigenmaps and Local Linear Embedding methods, is not complete. Approximating complex relationship as pairwise will lead to the loss of information. Let us consider classifying a set of genes into different gene functions. From [9], we may construct an un-directed graph in which the vertices represent the genes and two genes are connected by an edge if these two genes show a similar pattern of expression (i.e. the gene expression data is used as the datasets in [9]). Any two genes connected by an edge tend to have similar functions. However, assuming the pairwise relationship between genes is not complete, the information a group of genes that show very similar patterns of expression and tend to have similar functions [10] (i.e. the functional modules) is missed. The natural way overcoming the information loss is to represent the gene expression data as the hypergraph [10]. A hypergraph is a graph in which an edge (i.e. a hyper-edge) can connect more than two vertices. However, representing the dataset as the hypergraph will not lead to the perfection. The number of hyper-edges may be large; hence this will lead to high time complexity of the clustering methods or the classification methods when we try to apply the clustering/classification methods to this hypergraph dataset. Thus, there exists a need to develop the dimensional reduction methods for the hypergraph datasets. In [11,12], the symmetric normalized hypergraph Laplacian Eigenmaps has been developed and successfully applied to zoo dataset. However, the random walk and un-normalized hypergraph Laplacian Eigenmaps have not yet been developed and

applied to any practical applications. In this paper, we will develop the un-normalized hypergraph Laplacian Eigenmaps and apply this method combined with kernel ridge regression method to the zoo dataset available from UCI repository and the tiny version of the 20 newsgroups dataset.

Moreover, in the developed un-normalized hypergraph Laplacian Eigenmaps algorithm, we assume that the weights of all hyper-edges are equal to 1. This is not true at all in practical applications. In the other words, some hyper-edges may be more important than other hyper-edges and thus will have the weights that are larger than the weights of other hyper-edges. Thus, in this paper, we will also develop the weighted un-normalized hypergraph Laplacian Eigenmaps and apply this method combined with kernel ridge regression method to the zoo dataset available from UCI repository and the tiny version of the 20 newsgroups dataset.

We will organize the paper as follows: Section 2 will introduce the definitions of the three hypergraph Laplacians. Section 3 will present the un-normalized hypergraph Laplacian Eigenmaps algorithm in detail. Section 4 will present the weighted un-normalized hypergraph Laplacian Eigenmaps algorithm in detail. Section 5 will show the experimental results of the un-normalized hypergraph Laplacian Eigenmaps algorithm and the weighted un-normalized hypergraph Laplacian Eigenmaps algorithm combined with the kernel ridge regression method applied to the zoo dataset available from UCI repository and the tiny version of the 20 newsgroups dataset. Section 6 will conclude this paper and the future direction of researches will be discussed.

2 Preliminary notations and definitions

Given a hypergraph *G=(V,E)*, where *V* is the set of vertices and *E* is the set of hyper-edges. Each hyper-edge $e \in E$ is the subset of *V*. Please note that the cardinality of *e* is greater than or equal two. In the other words, $|e| \geq 2$, for every $e \in E$. Let *w(e)* be the weight of the hyper-edge *e*. Then *W* will be the $R^{|E|*|E|}$ diagonal matrix containing the weights of all hyper-edges in its diagonal entries.

The incidence matrix *H* of *G* is a $R^{|V|*|E|}$ matrix that can be defined as follows

$$h(v,e) = \begin{cases} 1 \ if \ vertex \ v \ belongs \ to \ hyperedge \ e \\ 0 \ otherwise \end{cases} \quad (1)$$

From the above definition, we can define the degree of vertex *v* and the degree of hyper-edge *e* as follows

$$d(v) = \sum_{e \in E} w(e) * h(v,e) \quad (2)$$

$$d(e) = \sum_{v \in V} h(v,e) \quad (3)$$

Let $D_v \ and \ D_e$ be two diagonal matrices containing the degrees of vertices and the degrees of hyper-edges in their diagonal entries respectively. Please note that D_v is the $R^{|V|*|V|}$ matrix and D_e is the $R^{|E|*|E|}$ matrix.

Please note that, we assume that the weight of each hyper-edge is 1.

The un-normalized hypergraph Laplacian [10, 11, 12] is defined as follows

$$L = D_v - HWD_e^{-1}H^T \quad (4)$$

The symmetric normalized hypergraph Laplacian [11, 12] is defined as follows

$$L_{sym} = I - D_v^{-\frac{1}{2}} HWD_e^{-1}H^T D_v^{-\frac{1}{2}} \quad (5)$$

The random walk hypergraph Laplacian [10, 11, 12] is defined as follows

$$L_{rw} = I - D_v^{-1} HWD_e^{-1}H^T \quad (6)$$

3 Un-normalized hypergraph Laplacian Eigenmaps algorithm

Suppose that we are given the hypergraph. In the other words, we know the topology of the hypergraph. What we want to do is to transform each node of the hypergraph to a numerical vector utilizing the topology of the hypergraph. The un-normalized hypergraph Laplacian Eigenmaps algorithm will exactly map each node of the hypergraph to numerical vector.

Un-normalized hypergraph Laplacian Eigenmaps algorithm

In this part, we will give the brief overview of the un-normalized hypergraph Laplacian Eigenmaps algorithm. The outline of this algorithm is as follows

1. Construct $D_v \ and \ D_e$ from the incidence matrix *H* of *G*

2. Compute the un-normalized hypergraph Laplacian $L = D_v - HWD_e^{-1}H^T$
3. Compute all eigenvalues and eigenvectors of L and sort all eigenvalues and their corresponding eigenvector in ascending order. Pick the first k eigenvectors $v_2, v_3, \ldots, v_{k+1}$ of L in the sorted list. k can be determined in the following two ways:
    a. k is the number such that $\frac{\lambda_{k+2}}{\lambda_{k+1}}$ is largest for all $2 \leq k \leq |V|$
    b. k is the number such that $\lambda_{k+2} - \lambda_{k+1}$ is largest for all $2 \leq k \leq |V|$
4. Let $U \in R^{|V| * k}$ be the matrix containing the vectors $v_2, v_3, \ldots, v_{k+1}$ as columns and U is the final result

4 Weighted un-normalized hypergraph Laplacian Eigenmaps algorithm

In the above un-normalized hypergraph Laplacian Eigenmaps algorithm, we assume that the weights of all hyper-edges are equal to 1. This is not true at all in practical applications. In the other words, some hyper-edges may be more important than other hyper-edges and thus will have the weights that are larger than the weights of other hyper-edges.

Hence in order to assign weights to hyper-edges of the hypergraph, to transform the nodes of the hypergraph to numerical vectors, and to improve the accuracy of the classification algorithms of hypergraphs, we would like to solve the following minimization problems

$$argmin_{f,w} \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \subseteq E} \frac{w(e)}{d(e)} (f(u) - f(v))^2 + \alpha ||f - y||^2 + \beta ||w||^2 \; such \; that \; 1_{|E|}^T w = 1 \quad (7)$$

Please note that w is the vector containing the weights of all the hyper-edges of the hypergraph, $f \in R^{|V|}$ is the final ranking vector (i.e. the output vector) that is used for the classification of the nodes of the hypergraph with some threshold value.

Moreover, we know that $\frac{1}{2}\sum_{e\in E}\sum_{\{u,v\}\subseteq E}\frac{w(e)}{d(e)}(f(u)-f(v))^2 = f^T Lf$ (8)

The proof of (8) can be found in [10].

Thus, we need to solve the following optimization problem

$$argmin_{f,w} f^T Lf + \alpha||f-y||^2 + \beta||w||^2 \; such\; that\; 1_{|E|}^T w = 1 \quad (9)$$

With a fixed *w*, we can optimize *f* like the following

$$argmin_f f^T Lf + \alpha||f-y||^2 = argmin_f f^T Lf + \alpha(f-y)^T(f-y))$$

In the other words, we need to solve

$$\frac{\partial(f^T Lf + \alpha(f-y)^T(f-y))}{\partial f} = 0$$

This will lead to

$$Lf + \alpha(f-y) = 0$$

$$(L+\alpha I)f = \alpha y$$

Hence the solution f^* of the above equations is

$$f^* = \alpha(L+\alpha I)^{-1}y$$

With a fixed *f*, we can optimize *w* like the following

$$argmin_w f^T Lf + \beta||w||^2 \; such\; that\; 1_{|E|}^T w = 1$$

The Lagrangian function of the above optimization problem is

$$\delta(w,\gamma) = f^T Lf + \beta w^T w + \gamma(1_{|E|}^T w - 1)$$

$$= f^T(D_v - HWD_e^{-1}H^T)f + \beta w^T w + \gamma(1_{|E|}^T w - 1)$$

The partial derivative of $\delta(w,\gamma)$ with respect to w_i is given by

$$\frac{\partial\delta}{\partial w_i} = \frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f + \beta w^T w + \gamma(1_{|E|}^T w - 1))}{\partial w_i}$$

$$= \frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} + 2\beta w_i + \gamma$$

Next, we need to solve

$$\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} + 2\beta w_i + \gamma = 0$$

This will lead to

$$w_i = \frac{1}{2\beta}\left[-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} - \gamma\right]$$

Moreover, we know that

$$1_{|E|}^{T} w = 1$$

In the other words, we have

$$\sum_i \frac{1}{2\beta}[-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} - \gamma] = 1$$

$$\frac{1}{2\beta}\sum_i [-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} - \gamma] = 1$$

$$2\beta = \sum_i (-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i}) - |E|\gamma$$

$$\gamma = \frac{1}{|E|}[\sum_i (-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i}) - 2\beta]$$

In general, we have

$$\gamma = \frac{1}{|E|}[\sum_i (-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i}) - 2\beta]$$

$$w_i = \frac{1}{2\beta}[-\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} - \gamma]$$

Now, we need to compute $\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i}$.

We have

$$\frac{\partial(f^T(D_v - HWD_e^{-1}H^T)f)}{\partial w_i} = f^T \frac{\partial(D_v)}{\partial w_i} f - f^T H \frac{\partial W}{\partial w_i} D_e^{-1} H^T f$$

$$= f^T(diag(H(:,i)) - D_e^{-1}(i,i)H(:,i)H(:,i)^T)f$$

Thus, finally, we have that

$$\gamma = \frac{1}{|E|}[\sum_i (-f^T(diag(H(:,i)) - D_e^{-1}(i,i)H(:,i)H(:,i)^T)f) - 2\beta] \quad (10)$$

$$w_i = \frac{1}{2\beta}[-f^T(diag(H(:,i)) - D_e^{-1}(i,i)H(:,i)H(:,i)^T)f - \gamma] \quad (11)$$

Weighted un-normalized hypergraph Laplacian Eigenmap algorithm

In this part, we will give the brief overview of the weighted un-normalized hypergraph Laplacian Eigenmaps algorithm. The outline of this algorithm is as follows

1. Construct D_v *and* D_e from the incidence matrix *H* and matrix *W* of *G* (initially, *W* is the identity matrix)
2. Compute the un-normalized hypergraph Laplacian $L = D_v - HWD_e^{-1}H^T$

3. Compute $f = \alpha(L + \alpha I)^{-1} y$
4. Compute the weight matrix W

$$\gamma = \frac{1}{|E|}[\sum_{i}(-f^T(diag(H(:,i)) - D_e^{-1}(i,i)H(:,i)H(:,i)^T)f) - 2\beta]$$

$$w_i = \frac{1}{2\beta}[-f^T(diag(H(:,i)) - D_e^{-1}(i,i)H(:,i)H(:,i)^T)f - \gamma]$$

5. Update the matrix D_v and W
6. Repeat step 2-step 5 until convergence and get the final ranking vector f.
7. Compute all eigenvalues and eigenvectors of the "updated" L and sort all eigenvalues and their corresponding eigenvector in ascending order. Pick the first k eigenvectors $v_2, v_3, \ldots, v_{k+1}$ of L in the sorted list. k can be determined in the following two ways:
    a. k is the number such that $\frac{\lambda_{k+2}}{\lambda_{k+1}}$ is largest for all $2 \leq k \leq |V|$
    b. k is the number such that $\lambda_{k+2} - \lambda_{k+1}$ is largest for all $2 \leq k \leq |V|$
8. Let $U \in R^{|V| * k}$ be the matrix containing the vectors $v_2, v_3, \ldots, v_{k+1}$ as columns and U is the final result

Clearly, from the above algorithm, we recognize that the weighted un-normalized hyper-graph Laplacian Eigenmaps algorithm contains the weighted hyper-graph based semi-supervised learning method starting from step 1 to step 6.

5 Experiments and Results

5.1 Description of datasets

In this paper, we used the zoo data set and the tiny version of 20 newsgroups dataset which can be obtained from UCI repository and from http://www.cs.nyu.edu/~roweis/data.html respectively. The zoo data set contains 101 animals with 17 attributes. The attributes include hair, feathers, eggs, milk,

etc. The animals have been classified into 7 different classes. In this dataset, each attribute is the hyper-edge. The tiny version of the 20 newsgroups dataset contains the binary occurrence data for 100 words across 16242 postings. However, we just choose small subset of this tiny dataset containing 4000 postings in order to test our algorithms. In this dataset, each word is the hyper-edge. Thus, these two datasets are themselves the hypergraphs and we don't need to preprocess these two datasets.

5.2 Experiments and Results

In this section, we experiment with the above proposed un-normalized hypergraph Laplacian Eigenmaps and the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method [13], the hyper-graph based semi-supervised learning method [10,11,12], and the weighted hyper-graph based semi-supervised learning method applied directly to the zoo dataset and the tiny version of 20 newsgroups dataset in terms of accuracy performance measure. The accuracy performance measure Q is given as follows

$$Q = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

All experiments were implemented in Matlab 6.5 on virtual machine. The accuracy performance measures of the four above proposed methods are given in the following table 1 and table 2.

Table 1: **Accuracies** of the four proposed methods which are the un-normalized hypergraph Laplacian Eigenmaps and the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, the hyper-graph based semi-supervised learning method, and the weighted hyper-graph based semi-supervised learning method for the **zoo dataset**

Accuracy (%)			
Un-normalized hypergraph Laplacian Eigenmaps + Kernel ridge regression method	Weighted un-normalized hypergraph Laplacian Eigenmaps + Kernel ridge regression method	Hyper-graph based semi-supervised learning method	Weighted hyper-graph based semi-supervised learning method
95.77	95.77	90.48	98.64

The following figure 1 shows the **accuracies** of the four proposed methods which are the un-normalized hypergraph Laplacian Eigenmaps and the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, the hyper-graph based semi-supervised learning method, and the weighted hyper-graph based semi-supervised learning method for the **zoo dataset:**

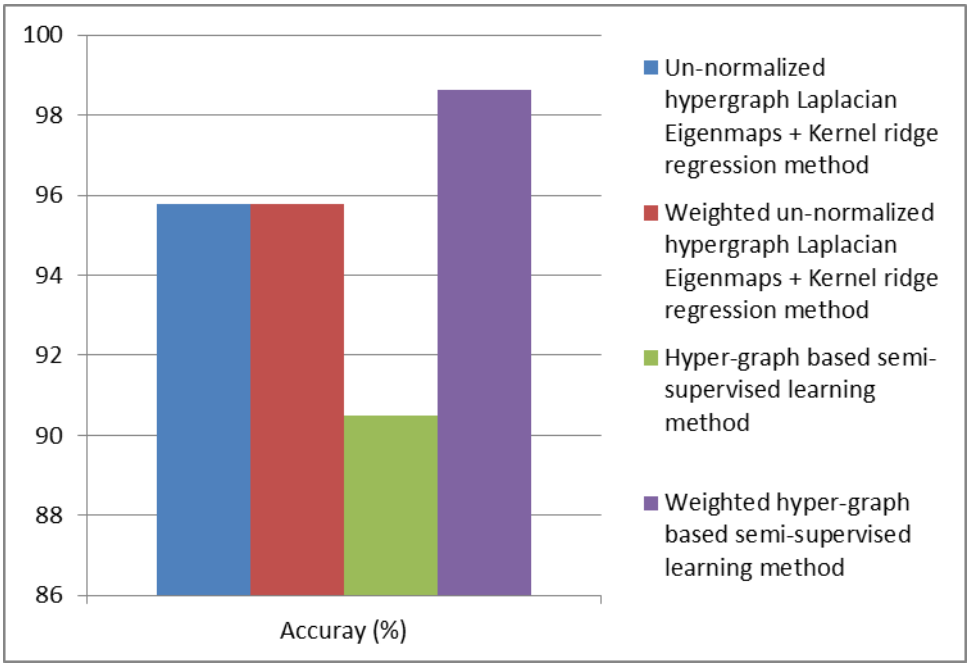
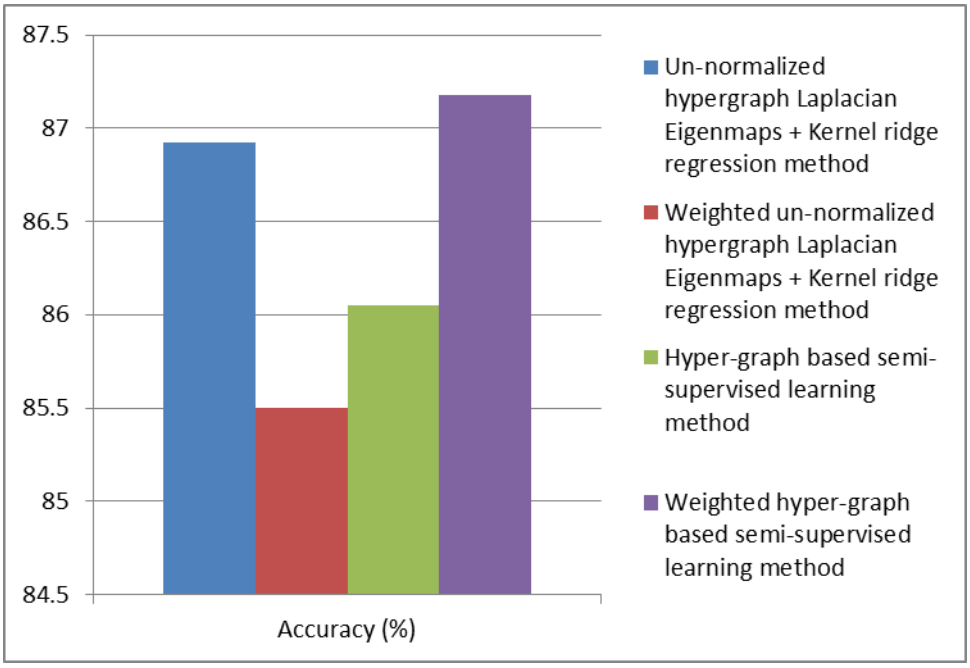


Table 2: **Accuracies** of the four proposed methods which are the un-normalized hypergraph Laplacian Eigenmaps and the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, the hyper-graph based semi-supervised learning method, and the weighted hyper-graph based semi-supervised learning method for the **20 newsgroups dataset**

Accuracy (%)			
Un-normalized hypergraph Laplacian Eigenmaps + Kernel ridge regression method	Weighted un-normalized hypergraph Laplacian Eigenmaps + Kernel ridge regression method	Hyper-graph based semi-supervised learning method	Weighted hyper-graph based semi-supervised learning method
86.92	85.50	86.05	87.18

The following figure 2 shows the **accuracies** of the four proposed methods which are the un-normalized hypergraph Laplacian Eigenmaps and the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, the hyper-graph based semi-supervised learning method, and the weighted hyper-graph based semi-supervised learning method for the **20 newsgroups dataset:**



5.3 Discussions

From the above figures, we easily recognized that the weighted hyper-graph semi-supervised learning method achieves the highest accuracy performance measures since the solution vector of the weighted hyper-graph semi-supervised learning method is directly obtained from the optimization problem (7) which is used to maximize the accuracy of the hypergraph semi-supervised learning method. In the other hands, the weighted un-normalized hypergraph Laplacian Eigenmaps do not always perform better the un-normalized hypergraph Laplacian Eigenmaps algorithm since these two algorithms are combined with the kernel ridge regression algorithm, which is not the best supervised classification algorithm (especially for the feature vectors which are results of the weighted un-normalized hypergraph Laplacian Eigenmaps and un-normalized hypergraph Laplacian Eigenmaps) for the scope of this paper.

6 Conclusion

In this paper, we have proposed the detailed algorithms the un-normalized hypergraph Laplacian Eigenmaps, the weighted un-normalized hypergraph Laplacian Eigenmaps applying to the zoo dataset and the tiny version of 20 newsgroups dataset. Interestingly, experiments show that the weighted un-normalized hypergraph Laplacian Eigenmaps algorithm do not always perform better the un-normalized hypergraph Laplacian Eigenmaps algorithm. However, the weighted hypergraph semi-supervised learning method do always perform better than the un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, the weighted un-normalized hypergraph Laplacian Eigenmaps combined with the kernel ridge regression method, and the hypergraph semi-supervised learning method.

In the future, the un-normalized hypergraph Laplacian Eigenmaps, the weighted un-normalized hypergraph Laplacian Eigenmaps, and the weighted hypergraph semi-supervised learning method will be applied to larger hypergraphs such as web hypergraph (to detect spam or not) and will be implemented in Python and MapReduce.

To the best of our knowledge, the un-normalized hypergraph p-Laplacian Eigenmaps has not yet been developed. This method is worth investigated because of its difficult nature and its close connection to partial differential equation on hypergraph field.

ACKNOWLEDGEMENTS.
This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number B2018-20-07.

References

[1] van der Maaten, Laurens JP, Eric O. Postma, and H. Jaap van den Herik. "Dimensionality reduction: A comparative review." *Journal of Machine Learning Research* 10.1-41 (2009): 66-71.

[2] Trang, Hoang, Tran Hoang Loc, and Huynh Bui Hoang Nam. "Proposed combination of PCA and MFCC feature extraction in speech recognition system." *Advanced Technologies for Communications (ATC), 2014 International Conference on*. IEEE, 2014.

[3] Turk, Matthew, and Alex P. Pentland. "Face recognition using eigenfaces." *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 1991.

[4] Vert, Jean-Philippe, and Yoshihiro Yamanishi. "Supervised graph inference." *Advances in Neural Information Processing Systems*. 2004.

[5] Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem." *Neural computation* 10.5 (1998): 1299-1319.

[6] Tenenbaum, Joshua B., Vin De Silva, and John C. Langford. "A global geometric framework for nonlinear dimensionality reduction." *Science* 290.5500 (2000): 2319-2323.

[7] Roweis, Sam T., and Lawrence K. Saul. "Nonlinear dimensionality reduction by locally linear embedding." *Science* 290.5500 (2000): 2323-2326.

[8] Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." *Neural computation* 15.6 (2003): 1373-1396.

[9] Tran, Loc. "Application of three graph Laplacian based semi-supervised learning methods to protein function prediction problem." *arXiv preprint arXiv:1211.4289* (2012).

[10] Tran, Loc. "Hypergraph and protein function prediction with gene expression data." *arXiv preprint arXiv:1212.0388* (2012).

[11] Zhou, Dengyong, Jiayuan Huang, and Bernhard Schölkopf. "Learning with hypergraphs: Clustering, classification, and embedding." *Advances in neural information processing systems*. 2006.

[12] Zhou, Dengyong, Jiayuan Huang, and Bernhard Scholkopf. "Beyond pairwise classification and clustering using hypergraphs." *Proceedings of the Neural Information Processing Systems*. 2005.

[13] Trang, Hoang, and Loc Tran. "Kernel ridge regression method applied to speech recognition problem: A novel approach." *Advanced Technologies for Communications (ATC), 2014 International Conference on*. IEEE, 2014.