Graph Convolutional Extreme Learning Machine

Zijia Zhang*, Yaoming Cai*, Wenyin Gong*†, Xiaobo Liu‡§, Zhihua Cai*
*School of Computer Science, China University of Geosciences, Wuhan, China
‡School of Automation, China University of Geosciences, Wuhan, China
§Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan, China
Emails: {zhangzijia, caiyaom, wygong, zhcai, xbliu}@cug.edu.cn

***Abstract*—Extreme Learning Machine (ELM) has gained lots of research interest due to its universal approximation capability and fast learning speed. However, traditional ELMs are devised for regular Euclidean data, such as 2D grid and 1D sequence, and thus don't apply to non-Euclidean data, e.g., graph-structured data. To overcome this shortcoming, this paper presents a Graph Convolutional Extreme Learning Machine (termed as GCELM) for semi-supervised classification. Technically, a random graph convolutional layer is introduced to replace the random projection of original ELM, which endues ELM with the capability of dealing with graph-structured data directly. To generate a robust graph from the raw dataset, a self-representation model is adopted to construct a weighted graph. Extensive experiments on 27 UCI datasets demonstrate that GCELM outperforms many popular semi-supervised methods, and with faster learning speed. To the best of our knowledge, this is the first work that combines graph convolution with ELM.**

***Index Terms*—Extreme Learning Machine, Graph Convolution Network, Semi-supervised Classification**

I. Introduction

Extreme Learning Machine (ELM) [1], [2], [3], a special case of random vector functional-link network (RVFL) [4], [5], is a single hidden layer feedforward neural network (SLFN) in which the hidden layer parameters are generated randomly and the output weights are calculated as a closed-form solution. ELM algorithms are characterized by a very light computational burden, since the training of the hidden layer is avoided [6]. Over the last decade, ELM has achieved great success in a variety of fields ranging from medical/biomedical data analysis [7], computer vision [8], [9], image processing [10], [11], [12], to system modeling and prediction [13], [14].

Despite promising performance and extensive studies, ELMs can only operate on regular Euclidean data, such as text (1D sequence) and images (2D grid). These data structures can be treated as special cases of non-Euclidean data. As a typical non-Euclidean data structure, graph-structured data is widely used to analyze the complex relationship between objects, e.g., social network [15] and molecule [16]. However, how to apply ELM to graph-structured data is still an open problem that fails to draw too much attention. This motivates us to extend the classical ELM into the non-Euclidean domain and enable it to learn on the graph. We refer to this kind of ELM as Graph Convolutional Extreme Learning Machine (GCELM).

† W. Gong is the corresponding author.

Recently, Graph Neural Network (GNN) [17], [18] has gained increasing research interests due to its powerful ability for representation learning on graph-structured data. Unlike traditional neural networks, GNNs capture the dependence of graphs via message passing between the nodes of graphs. Specifically, GNNs gather information from each node's neighbors and update nodes' hidden states. By analogy with convolutional neural network (CNN) [19], convolutions in graph domain have drawn lots of interest. GNNs equipped with generalized convolutions are often referred to as Graph convolutional network (GCN) [20]. Since spectral representation of graphs can provide a theoretical guarantee, it is frequently used as an alternative for the implementation of graph convolution.

Compared with traditional neural networks, the propagation step of GCN can be regarded as a fully connected layer that cooperated with Laplacian smooth [21]. Depending on this point of view, we introduce graph convolution into ELM to replace its hidden layer, which endues it with the capacity of processing graph-structured data. Following the common procedure of ELM, we randomly generate filter parameters of graph convolution and calculate ELM's output weights as a closed-form solution. This reserves the main advantages of ELMs according to the theory of ELMs. That is fast learning speed and universal approximation capability. Furthermore, our proposed method can be used as a semi-supervised classification approach, since it takes unlabeled data into account. Nevertheless, it is different from semi-supervised ELM (SS-ELM) [22] because SS-ELM incorporates structure information in the output layer by casting it as a manifold regularization term.

To sum up, the main contributions of this paper are:

1) We propose a graph convolutional ELM, i.e., GCELM, for semi-supervised classification. GCELM is able to learn on graph-structured data directly and keeps the advantages of GCN and ELM. To the best of our knowledge, this is the first work that combines ELM with graph convolution. The successful attempt signifies that considering structure information among data is important for ELM, which offers an alternative orientation for graph representation learning.
2) A robust graph construction method is employed to fast generate graph information from raw data. The

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

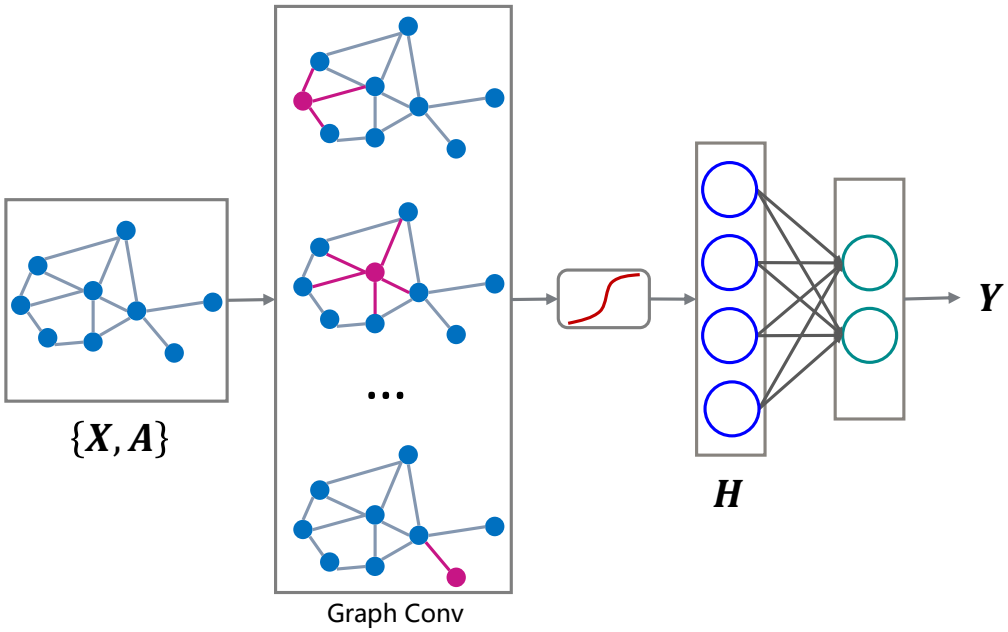


Fig. 1. Overview of the proposed GCELM. The method takes data features and its graph structure as inputs, and consists of a random graph convolutional layer and a ELM classifier layer.

method models the intrinsic structure of data using self-representation, and thus is robust to noise. By slacking self-representation with Frobenius regularization, the affinity matrix (graph edge weights) can be expressed as a closed-form solution. This keeps consistency with ELM and accelerates learning.

3) We conduct extensive experiments on many popular datasets for semi-supervised classification tasks. Comparisons with state-of-the-art methods demonstrate the effectiveness of the proposed GCELM. Code is available at: https://github.com/AngryCai/GCELM.

This paper is structured as follows. Related works on ELM and GCN are presented in Section II. The proposed graph convolutional ELM and robust graph generation are introduced in Section III. Experimental results are presented in Section IV. Finally, Section V concludes the paper with a brief summary.

II. Related Work

In this section, we give a brief review of the basic ELM and its semi-supervised version, and the recent progress of GCN.

A. Basic ELM and Semi-Supervised ELM

The basic ELM can be interpreted as two components, i.e., random generated hidden layer and closed-form output layer. Formally, in the first stage, ELM's hidden layer can be expressed as

$$\boldsymbol{H} = \sigma\left(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}\right), \tag{1}$$

where $\boldsymbol{H}$ is the hidden layer output matrix that takes $\boldsymbol{X}$ as input. The hidden layer is parameterized by hidden weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$, both are generated randomly. σ denotes nonlinear activation function such as Sigmoid and Tanh. In the second stage, ELM computes prediction as

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{\beta}. \tag{2}$$

Here, $\boldsymbol{Y}$ is the prediction of ELM and $\boldsymbol{\beta}$ is the output weights matrix. Since $\boldsymbol{H}$ is known to the output layer, Eq. (2) essentially is a least squares optimization problem and can be solved as

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{Y}, \tag{3}$$

where $\boldsymbol{H}^{\dagger}$ indicates Moore–Penrose generalized inverse of matrix $\boldsymbol{H}$. ELM avoids iteratively parameters tuning, thus it is significantly faster than gradient based neural networks.

To enhance the basic ELM, numerous ELM variants have been proposed, including kernel ELM [23], evolutionary ELM [24], deep ELM [3], and so on. To leverage the unlabelled data, some works devoted to extending ELM to semi-supervised task. The representative semi-supervised ELM (SS-ELM) [22] is based on manifold regularization, usually denoted as $Trace\left(\boldsymbol{\beta}^T\boldsymbol{H}_{all}^T\boldsymbol{L}\boldsymbol{H}_{all}\boldsymbol{\beta}\right)$, where $\boldsymbol{H}_{all}$ represents the hidden output of all samples (labeled and unlabelled) and $\boldsymbol{L}$ denotes Laplacian matrix. Thus, the optimization problem of SS-ELM can be written as

$$min\left\|\boldsymbol{H}_{labeled}\boldsymbol{\beta} - \boldsymbol{Y}_{labeled}\right\|_F^2 + Trace\left(\boldsymbol{\beta}^T\boldsymbol{H}_{all}^T\boldsymbol{L}\boldsymbol{H}_{all}\boldsymbol{\beta}\right), \tag{4}$$

where $\boldsymbol{H}_{labeled}$ denotes labeled hidden outputs and $\boldsymbol{Y}_{labeled}$ is the training labels.

B. Graph Convolutional Networks

Graph convolutional networks (GCNs) generalize traditional convolutional neural networks to the graph domain. It has been proven to be excellent for graph representation learning. GCNs can be categorized into two schemes, i.e., spectral GCNs and spatial GCNs. Here, we focus only on the first GCN [20]. Let $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be the adjacency matrix of an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The propagation rule of one simplified GCN model can be written as

$$\boldsymbol{H} = \sigma\left(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{W}\right), \tag{5}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{I}_{|\mathcal{V}|} + \boldsymbol{A}$ is the adjacency matrix considered self-connection and $\tilde{\boldsymbol{D}}$ denotes diagonal degree matrix of nodes whose elements given by $\tilde{D}_{ii} = \sum_j \tilde{A_{ij}}$. Eq. (5) is regarded as the graph convloutional layer, which can be optimized using gradient descent. By stacking several such layers, GCN can use to learn deep graph representation. However, many studies have proven that the performance of GCN will be degraded as the depth due to the overfitting issue.

III. Graph Convolutional ELM

The classical ELM can be treated as a nonlinear random transformation followed by a ridge regression classifier. However, the random transformation is sub-optimal and often neglects the graph-structured information, resulting in inaccurate feature transformation. Furthermore, such random transformation doesn't apply to the graph-structured data. To address this problem, we introduce the graph convolution into ELM. The overview of the proposed GCELM is given in Fig. 1 and more details are described in the following subsections.

A. Random Graph Convolution

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be an undirected graph that is composed of a node set $v_i \in \mathcal{V}$ with the size of N and an edge set $(v_i, v_j) \in \mathcal{E}$ with size $|\mathcal{E}|$. We indicate the adjacency matrix of $\mathcal{G}$ as $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ where each element A_{ij} can be binarily denoting whether v_i join with v_j, or a scalar signifying the weights of edge (v_i, v_j). Suppose each node contain a feature vector $x_i \in \mathbb{R}^m$ with a label $y_i \in \{1, 2, \cdots, C\}$, where C is the number of distinct classes. For clarity, we indicate feature set and label set in matrix forms, i.e., $\boldsymbol{X} \in \mathbb{R}^{N\times m}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times C}$, where $\boldsymbol{Y}$ is binarized label matrix using one-hot encoding.

Inspired by the recent development of graph neural networks, we introduce graph convolution into ELM. Specifically, instead of adopting simple random projection like the classical ELM, we extend ELM to the non-Euclidean domain by introducing random graph convolution. We define the random graph convolutional layer in ELM as

$$\boldsymbol{H} = \sigma(\mathcal{A}\boldsymbol{X}\boldsymbol{W}). \tag{6}$$

Here, $\boldsymbol{W} \in \mathbb{R}^{m\times L}$ denotes the hidden layer parameters and σ signifies the nonlinear activation function. $\mathcal{A}$ is the normalized adjacency matrix which is formulated by

$$\mathcal{A} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}, \tag{7}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ is the adjacency matrix with added self-connections, and $\tilde{D}$ denotes the diagonal degree matrix of $\tilde{\boldsymbol{A}}$ whose elements are given by $\tilde{D}_{ii} = \sum_j \tilde{A_{ij}}$. Inherited from the classical ELM, we generate $\boldsymbol{W}$ randomly according to a continuous probability distribution. For convenience, we omit biases but it is easy to add biases by augmenting X using one column with all elements 1.

Formally, compared with the original ELM hidden layer, the random graph convolutional layer adds an extra matrix operation, i.e., $\mathcal{A}$. This is the main difference between the random graph convolutional layer and the original ELM hidden layer and this also endues ELM with the capability of processing structure data.

B. Fast Semi-Supervised Node Classification

One of the main advantages of ELM is fast learning speed, which is benefited from its closed-form solution in the output layer. In this paper, we keep such advantage by using ELM classification layer as GCELM's output layer.

Let $\boldsymbol{X}_{\mathcal{T}} \in \mathbb{R}^{N_{\mathcal{T}}\times m}$ and $\boldsymbol{Y}_{\mathcal{T}} \in \mathbb{R}^{N_{\mathcal{T}}\times C}$ be the labeled nodes' feature set and their one-hot label matrix, where $N_{\mathcal{T}}$ denotes the number of labeled nodes. Further let $\boldsymbol{X}_{\mathcal{U}} \in \mathbb{R}^{N_{\mathcal{U}}\times m}$ be the unlabelled nodes' feature set, where $N_{\mathcal{U}}$ is the number of unlabelled nodes. To learn the output layer weights, denoted as $\boldsymbol{\beta} \in \mathbb{R}^{L\times C}$, we first divide the hidden layer outputs into labeled and unlabeled two parts, i.e., $\boldsymbol{H}_{\mathcal{T}}$ and $\boldsymbol{H}_{\mathcal{U}}$. Our goal is to assign certain labels for those unlabeled nodes. To this end, we formulate the objective function as

$$\boldsymbol{H}_{\mathcal{T}}\boldsymbol{\beta} = \boldsymbol{Y}_{\mathcal{T}}. \tag{8}$$

We solve Eq. (8) by reformulating it as following regularized ridge regression optimization problem

$$\arg\min_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\boldsymbol{H}_{\mathcal{T}}\boldsymbol{\beta} - \boldsymbol{Y}_{\mathcal{T}}\right\|_F^2 + \frac{\lambda}{2}\left\|\boldsymbol{\beta}\right\|_F^2. \tag{9}$$

Here, $\mathcal{L}$ denotes loss function. λ is a nonnegative regularization coefficient. Eq. (9) has a closed-form solution that can be obtained by computing the partial derivative of $\mathcal{L}$ with respect to $\boldsymbol{\beta}$. The partial derivative can be written as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \boldsymbol{H}_{\mathcal{T}}^T\boldsymbol{H}_{\mathcal{T}}\boldsymbol{\beta} + \lambda\boldsymbol{\beta} - \boldsymbol{H}_{\mathcal{T}}^T\boldsymbol{Y}_{\mathcal{T}}. \tag{10}$$

By setting Eq. (10) to zero, we obtain the solution to the GCELM as follow:

$$\boldsymbol{\beta} = \left(\boldsymbol{H}_{\mathcal{T}}^T\boldsymbol{H}_{\mathcal{T}} + \lambda\boldsymbol{I}_L\right)^{-1}\boldsymbol{H}_{\mathcal{T}}^T\boldsymbol{Y}_{\mathcal{T}}. \tag{11}$$

Therefore, the labels of those unlabeled nodes can be determined by

$$\boldsymbol{Y}_{\mathcal{U}} = \boldsymbol{H}_{\mathcal{U}}\boldsymbol{\beta}. \tag{12}$$

C. Robust Graph Generation

In this subsection, we describe how to generate a robust graph from raw data. Although many works have been proposed to exploit the structure information among data, such as the k-nearest neighbors graph [25], they are sensitive to noise. To remedy this problem, we propose using a self-representation model to construct a weighted graph. Formally, we define the self-representation as

$$\boldsymbol{X}^T\boldsymbol{Z} = \boldsymbol{X}^T, s.t., diag\left(\boldsymbol{Z}\right) = 0. \tag{13}$$

Here, $\boldsymbol{Z} \in \mathbb{R}^{N\times N}$ denotes the coefficient matrix in which each column reflects the contribution that a sample is linearly represented by the remaining samples. To avoid trivial solution, the diagonal elements $diag\left(\boldsymbol{Z}\right)$ are usually constrained to zeros. However, Pan et al. [26] have proven that this constraint is unnecessary when a Frobenius regularization is used. Following this corollary, we solve Eq. (13) by minimizing the following objective

$$\arg\min_{\boldsymbol{Z}} \frac{1}{2}\left\|\boldsymbol{X}^T\boldsymbol{Z} - \boldsymbol{X}^T\right\|_F^2 + \frac{\alpha}{2}\left\|\boldsymbol{Z}\right\|_F^2. \tag{14}$$

Here, α denotes regularization coefficient. Similar to ELM, this objective has a closed-form solution that can be expressed as

$$\boldsymbol{Z} = \left(\boldsymbol{X}\boldsymbol{X}^T + \alpha\boldsymbol{I}_N\right)^{-1}\boldsymbol{X}\boldsymbol{X}^T. \tag{15}$$

Subsequently, we use $\boldsymbol{Z}$ as the estimation of edge weights and construct the adjacency matrix as

$$\boldsymbol{A} = \frac{1}{2}\left(\left|\boldsymbol{Z}\right| + \left|\boldsymbol{Z}^T\right|\right). \tag{16}$$

It is noteworthy that this method leads to a dense connected graph rather than sparse connected graph like k-nearest neighbors graph. The abovementioned graph generation method is also known as subspace learning [27] due to it essentially exploits the intrinsic subspace structure of data.

Algorithm 1: GCELM

Input: $\boldsymbol{X}_\mathcal{T}$, $\boldsymbol{X}_\mathcal{U}$, $\boldsymbol{Y}_\mathcal{T}$, α and λ
Output: $\boldsymbol{Y}_\mathcal{U}$

1 Construct robust graph adjacency matrix $\boldsymbol{A}$ by Eq. (16);
2 Calculate normalized adjacency matrix $\mathcal{A}$ by Eq. (7);
3 Initialize hidden layer parameters $\boldsymbol{W}$;
4 Calculate graph convolution outputs $\boldsymbol{H} = \sigma\left(\mathcal{A}\boldsymbol{X}\boldsymbol{W}\right)$;
5 Calculate output layer parameters $\boldsymbol{\beta}$ by Eq. (11);
6 Predict unlabelled samples' labels $\boldsymbol{Y}_\mathcal{U}$ via (12);

D. Remarks

Having introduced graph convolution into ELM, we endue classical ELM with the capacity of leaning on graph. We give the overall procedures of the proposed GCELMs in Algorithm 1. It can be seen that there is no iterative operation in the GCELMs. Furthermore, all the key steps of GCELMs can be represented as closed-form. Therefore, GCELMs retain the main advantage of ELMs, i.e., high-efficiency training, which makes GCELM easy implementation. This is different from GCN [20] which needs great effort to parameters tuning. In other words, GCELM will be much faster than GCN theoretically.

Comparing with SS-ELM [22], the main difference between them is that our method combines graph structure information in the hidden layer, while SS-ELM utilizes graph information by adding an additional Laplacian regularization term into the objective function. This signifies that SS-ELM is difficult to directly process structured data. Moreover, GCELM uses graph convolution operation in the hidden layer enabling structured information to be embedded into the intermediate process. In fact, one can adopt deep ELM to generate deeper graph representation.

IV. Experiments

A. Data Sets and Setup

We evaluate our method on 27 widely used classification datasets that are taken from University of California at Irvine (UCI) repository [1]. These datasets include binary-class and multi-class classification tasks. All the datasets are scaled into the range $[0, 1]$ using the min-max standardization technique. For each dataset, we randomly take 5 samples from each class as the labeled sample set and the rest as the unlabelled sample set.

We compare our method with seven baselines, i.e., basic ELM [2], KELM [28], SS-ELM [22], Transductive Support Vector Machine (TSVM) [29], Self-training Semi-supervised ELM (ST-ELM) [30], Laplacian Support Vector Machine (LapSVM) [31], and GCN [20]. We set 100 hidden neurons for the methods that contain hidden layers, namely GCELM, ELM, SS-ELM, ST-ELM, and GCN. The hyper-parameters involved in different methods are determined by grid search. All the methods are implemented with Python 3.5 running on an Intel i5-6500 3.20 GHz CPU with 8.00GB RAM.

B. Performance Comparison

Table I shows the comparative accuracy obtained by different methods on 27 datasets. All the results are calculated by averaging the results from 30 independent runs. In Table I, the field marked with • and ◦ denote that GCELM's classification accuracy is statistically and significantly better or worse than the method shown in the corresponding column. While the filed without mark signifies that there is no significant difference between GCELM and the corresponding method. We determine the significance by conducting a paired two-tailed t-tests with significance level $p = 0.05$ [3]. Basing on the significance testing results, we summarize the Win/Tie/Lose (**W/T/L**) values at the bottom of the table, where **W/T/L** indicates that, compared to their competitors, GCELM won on **W** datasets, tied on **T** datasets, and lost on **L** datasets. In

[1] https://archive.ics.uci.edu/ml/datasets.php

TABLE I
AVERAGE TESTING ACCURACY OF DIFFERENT METHODS (MEAN±STD, BEST IN BOLD)

Data sets	GCELM	SS-ELM	ELM	KELM	TSVM	ST-ELM	LapSVM	GCN
austra	**80.78±2.99**	76.24±8.14•	73.32±10.48•	79.47±4.12	77.43±10.59	75.17±9.99•	55.70±0.21•	71.38±8.36•
australian	**78.46±4.82**	78.10±6.24	71.35±11.25•	79.32±6.11	75.38±10.81	78.57±9.56	75.79±6.61	75.36±7.56
breast	61.61±8.61	55.30±7.67•	52.00±7.38•	55.56±8.97•	55.77±6.78•	53.18±10.68•	**71.54±0.00**○	60.17±6.18
cleve	**72.90±6.42**	72.88±5.30	68.95±7.31•	72.38±3.79	72.80±5.70	68.85±8.30•	54.20±0.00•	70.17±6.14
diabetes	**70.10±3.09**	66.27±8.92•	57.31±8.51•	65.71±5.64•	64.97±5.26	58.43±8.56•	65.30±0.06•	62.93±6.62•
dnatest	**67.87±5.03**	51.60±4.63•	50.77±4.79•	50.81±4.26•	60.99±2.94•	42.86±8.51•	48.27±4.10•	59.33±3.95•
german	67.10±10.01	54.48±5.35•	57.95±9.42•	51.07±7.48•	59.52±6.89•	59.64±7.63•	**70.21±0.03**	59.54±5.32•
heart	**76.53±3.63**	72.54±6.06•	69.15±9.07•	73.25±8.99	70.23±7.77•	67.21±8.62•	55.77±0.00•	70.00±8.58•
ionosphere	**81.07±7.51**	75.98±4.31•	74.65±8.96•	75.57±6.18•	77.05±5.43•	77.20±5.68•	64.52±0.00•	71.32±8.21•
iris	**94.32±3.20**	80.04±4.79•	80.70±10.76•	93.04±3.88	93.00±3.70	81.04±8.36•	92.70±2.58•	91.19±3.61•
sonar	**67.44±6.64**	63.28±5.10•	64.66±6.37•	62.56±5.37•	66.41±5.27	62.21±5.10•	46.46±0.00•	67.22±7.09
vote	84.19±5.72	83.75±3.49	84.39±6.17	87.98±1.97○	87.97±3.59○	**87.65±6.21**○	38.35±0.00•	86.62±3.87
WBC	**96.53±0.74**	92.30±1.83•	89.35±5.63•	95.30±2.78•	95.40±1.90•	92.14±3.50•	65.26±0.07•	95.93±2.28
weather	**76.94±14.38**	69.72±15.14	73.61±12.74	69.17±11.21•	76.39±14.29	70.00±15.00	41.67±0.00•	50.83±11.66•
Wine	**91.25±2.28**	89.37±3.33•	79.75±6.72•	90.31±3.92	90.51±3.34	79.59±7.68•	88.45±1.60•	90.84±2.79
X8D5K	**100.00±0.00**	**100.00±0.00**	96.93±3.07•	99.96±0.06•	99.98±0.05•	99.93±0.21	**100.00±0.00**	**100.00±0.00**
zoo	99.22±1.32	98.02±1.45	98.18±1.51•	98.65±1.60	99.48±1.09	99.69±0.94	99.95±0.28○	**100.00±0.00**○
cloud	**90.18±3.68**	89.92±3.22	73.38±10.32•	88.48±4.36	89.71±4.65	75.95±9.54•	81.32±4.86•	88.10±3.90•
bupa	**56.33±6.93**	52.55±4.01•	54.07±6.63	56.15±4.98	55.95±5.62	55.41±5.85	41.79±0.00•	51.72±3.71•
air	71.30±6.25	66.52±4.50•	67.44±5.83•	70.89±4.18	70.68±6.38	69.53±7.55	72.49±7.63	**79.65±5.78**○
segmentation	**83.22±3.14**	78.91±3.42•	65.71±7.74•	83.01±3.22	81.85±4.53	67.05±8.36•	80.84±3.51•	80.74±3.61•
pima In. D.	**68.45±5.55**	67.28±7.67	58.43±7.06•	63.52±6.78•	64.84±8.34	62.35±5.65•	35.29±1.03•	65.47±4.70•
Xinp	93.48±1.95	93.04±2.31	80.31±6.64•	92.76±3.00	93.90±1.76	81.56±8.59•	92.45±1.32•	**94.97±1.82**○
Normal7	99.08±0.42	84.85±1.02•	84.33±4.22•	99.45±0.24○	99.35±0.32○	84.86±7.06•	**99.58±0.12**○	95.26±1.12•
wdbc	**92.13±0.16**	88.94±4.59•	81.20±8.63•	91.72±4.20	91.15±4.02	89.51±5.39•	62.97±0.00•	90.86±2.44•
ecoli_label	64.46±7.16	**67.09±2.29**	57.92±11.95•	61.17±12.58	58.92±13.97	63.28±9.71	61.80±8.44	63.14±8.21
appendicitis	**85.83±2.80**	73.33±14.10•	58.12±11.00•	69.06±12.88•	63.70±15.29•	57.86±15.23•	16.67±0.00•	76.15±10.68•
Average	80.4±4.61	75.64±5.14	71.26±7.78	76.9±5.29	77.53±5.94	72.62±7.68	65.9±1.57	76.63±5.12
W/T/L	–	17/10/0	24/3/0	11/14/2	8/17/2	19/7/1	19/5/3	15/9/3

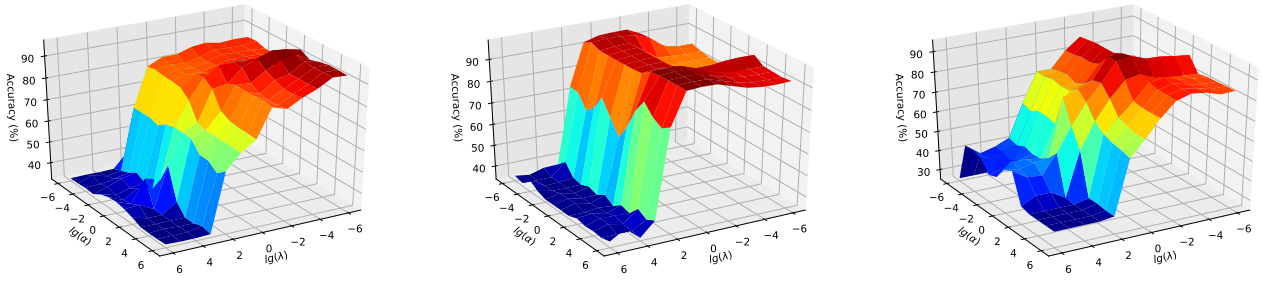
addition, the arithmetic mean accuracy of each method over all the datasets is given at the bottom of the table.

It can be seen that GCELM achieves the best accuracy on 18 datasets. The arithmetic mean accuracy of GCELM is 80.40% that is 2.87% better than the second better baseline, TSVM (77.53%). A more detailed analysis is given bellow.

- **GCELM v.s. GCN:** Comparing with GCN, our method is significantly better on 16 datasets and achieves similar performance on 9 datasets, although GCN has updated its parameters for 100 epoch with learning rate of 0.002. This signifies that 1) the random graph convolution operation is beneficial for semi-supervised classification; 2) the iterative parameters updating in GCN may be optional.
- **GCELM v.s. SS-ELM and ST-ELM:** GCELM wins on 17 and 19 datasets by comparing with SS-ELM and ST-ELM, respectively. Although these methods are ELM-based semi-supervised methods, GCELM and SS-ELM are based on the graph while ST-ELM relies on pseudo labels generated by self-training strategy. The comparative results with SS-ELM show that embedding graph information into feature representation is more efficient than the traditional manifold regularization.
- **GCELM v.s. ELM and KELM:** Comparing with the basic ELM and KELM, GCELM wins on 24 and 11 datasets, respectively. Since ELM uses only labeled samples, its classification accuracy is relatively lower than other semi-supervised methods. By utilizing kernel trick, KELM can achieve significant improvement but it is still not superior to our method.
- **GCELM v.s. TSVM and LapSVM:** Despite there are 17 datasets that have no statistical significance comparing TSVM with GCELM, GCELM achieves higher classification accuracy on most datasets. Especially, GCELM wins on 8 datasets which is much larger than that TSVM wins. Comparing with LapSVM, GCELM achieves better performance on 19 datasets, demonstrating the effectiveness and superiority of our method.

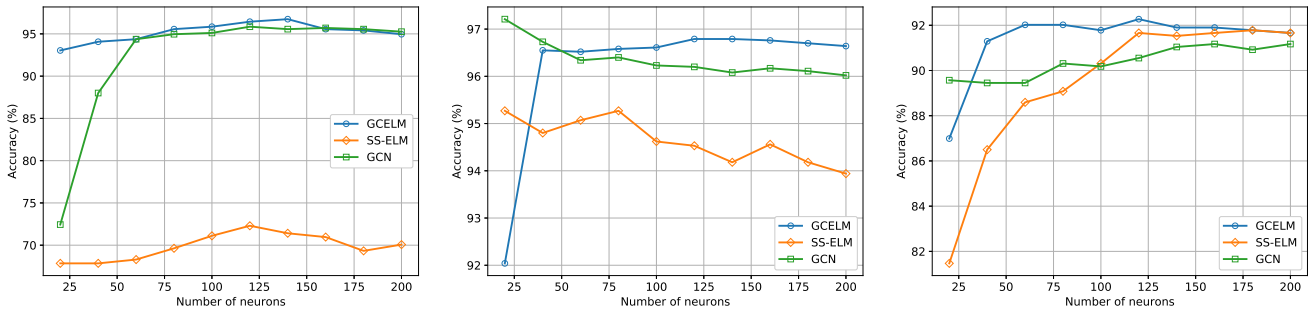
C. Sensitivity Analysis of Hyper-parameters

To observe the sensitivity of the main hyper-parameters of GCELM, i.e., λ, α, and the number of hidden neurons, we evaluate the performance of GCELM with varying hyper-parameter values. Fig. 2 (a)-(c) show the impact of λ and α for Iris, WBC, and Wine dataset. The two hyper-parameters vary in range from $1e^{-6}$ to $1e^{6}$ and are represented with logarithmic values. As can be seen, the classification accuracy of GCELM



(a) Iris (b) WBC (c) Wine

Fig. 2. Impact of λ and α where x-axis and y-axis denotes $\lg(\lambda)$ and $\lg(\alpha)$, respectively.



(a) Iris (b) WBC (c) Wine

Fig. 3. Performance comparison of SS-ELM, GCN, and our method with different hidden neurons.

tends towards increasing as λ decreased, while is insensitive to α. This indirectly demonstrates that the classification accuracy of GCELM depends mainly upon its output layer.

In Fig. 3, we show the influence of the number of hidden neurons involved in GCELM, SS-ELM, and GCN. We plot the average accuracy obtained by varying the number of hidden neurons from 20 to 200 with a interval of 20. It can be seen that, by increasing the number of hidden neurons, GCELM can achieve better classification accuracy, and furthermore, it is generally superior to SS-ELM and GCN with same setttings. It should be noted that the classification accuracy of SS-ELM and GCN might be degraded when using too many hidden neurons, e.g., WBC. That because more hidden neurons increase the risk to overfit the training data.

D. Running Time

In this experiment, we compare the running time of GCELM and all the baselines. The results are given in Table II. We can observe that GCELM is significantly faster than TSVM, ST-ELM, and GCN. Since GCELM includes more matrix operations than ELM, KELM, and SS-ELM, such as graph construction, it takes slightly more running time. ST-ELM and GCN contain iterative operations thus they commonly spent more time to update parameters. To sum up, GCELM keeps the advantage of fast learning speed existing in ELMs. This benefits from the closed-form solutions of output weights and graph generation.

V. Conclusions

We have presented a novel semi-supervised classification approach (GCELM) which generalizes ELM to graph-structured data. GCELM uses random graph convolutional layer to generate hidden representation and keeps fast learning speed by calculating output weights as close-form solution. To construct a robust graph structure from datasets, a self-representation model is adopted to estimate the similarity between data points. We evaluate GCELM on many popular datasets. The experimental results show that GCELM not only outperforms many semi-supervised classification models, especially GCN, but keeps the advantage of ELM, i.e., fast learning speed.

It should be noted that the proposed method is a preliminary work. We evaluated it on traditional non-structured data but it can be applied to graph-structured data. Furthermore, the proposed method offers an alternative orientation for ELM. Therefore, one can consider various useful strategies used in ELM into our method. For example, one can try to make random mapping more stable and make the model deeper. In our future works, we will further study the proposed method and give more extensive applications.

Acknowlegment

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61573324, 61873328, and 61973285, the National Natural

TABLE II
COMPARISON OF RUNNING TIME (IN SECOND).

Data sets	GCELM	SS-ELM	ELM	KELM	TSVM	ST-ELM	LapSVM	GCN
austra	0.0807	0.0333	0.0011	0.001	0.1761	0.8139	0.0802	0.6895
australian	0.0858	0.0341	0.0013	0.001	0.1816	0.7839	0.0799	0.7015
breast	0.0131	0.0119	0.0010	0.0010	0.0636	0.9096	0.0194	0.3354
cleve	0.0679	0.0129	0.0010	0.0010	0.0574	0.9843	0.0227	0.3447
diabetes	0.4853	0.0226	0.0012	0.001	0.4382	1.4124	0.1053	0.7454
dnatest	0.2332	0.0830	0.0033	0.0020	15.2089	1.6511	1.8055	1.7704
german	0.1403	0.0599	0.0020	0.0010	0.6302	1.4070	0.2303	1.1184
heart	0.0116	0.0135	0.0010	0.0010	0.0498	0.8333	0.0190	0.3452
ionosphere	0.0215	0.0087	0.0010	0.0010	0.1175	0.1150	0.0249	0.5574
iris	0.0053	0.0024	0.0010	0.0004	0.0585	0.0894	0.0256	0.3864
sonar	0.0099	0.0103	0.0010	0.0010	0.0921	0.4789	0.0206	0.3301
vote	0.0361	0.0189	0.0010	0.0010	0.0988	0.8711	0.0428	0.4789
WBC	0.0658	0.0279	0.0010	0.0010	0.0866	1.4105	0.0779	0.7195
weather	0.0033	0.0052	0.0010	0.0010	0.0084	0.0078	0.0059	0.2591
Wine	0.0062	0.0071	0.0007	0.0010	0.0700	0.7059	0.0632	0.3407
X8D5K	0.1435	0.0516	0.0020	0.0011	0.3224	1.5884	0.9441	1.1851
zoo	0.0059	0.0066	0.0010	0.0011	0.0358	0.1386	0.0441	0.2818
cloud	0.2160	0.0427	0.0022	0.0010	0.4260	1.5406	0.2120	1.1149
bupa	0.0174	0.0087	0.0010	0.0010	0.1712	1.1659	0.0233	0.3945
air	0.0221	0.0132	0.0010	0.0010	0.9576	1.1664	0.1753	0.4491
segmentation	0.0086	0.0087	0.0010	0.0011	0.2186	0.9132	0.1391	0.3295
pima Indians Ddiabetes	0.0859	0.0225	0.0011	0.0010	0.4453	1.4115	0.1063	0.7535
Xinp	0.0396	0.1538	0.0085	0.0052	0.1556	1.1651	0.8320	2.1210
Normal7	22.5832	7.1307	0.1521	0.0218	25.0076	15.3488	668.768	46.9471
wdbc	0.0476	0.1923	0.0161	0.0070	0.3248	2.0556	1.0375	2.2494
ecoli_label	0.1208	0.1924	0.0110	0.0060	0.1247	0.9238	0.3322	2.1403
appendicitis	0.0132	0.0240	0.0070	0.0058	0.0761	0.1568	0.0658	2.0623

Science Fund for Distinguished Young Scholars of China under Grant No. 61525404, and in part by the Fundamental Research Founds for Central Universities, China University of Geosciences (Wuhan) under Grant Nos. CUG160603 and 1910491T06.

REFERENCES

[1] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32 – 48, 2015.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[3] Y. Cai, X. Liu, Y. Zhang, and Z. Cai, "Hierarchical ensemble of extreme learning machine," *Pattern Recognition Letters*, 2018.

[4] Y. . Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, May 1992.

[5] Y. Zhang, J. Wu, Z. Cai, B. Du, and P. S. Yu, "An unsupervised parameter learning model for rvfl neural network," *Neural Networks*, vol. 112, pp. 85 – 97, 2019.

[6] C. Cervellera and D. Macció, "An extreme learning machine approach to density estimation problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3254–3265, Oct 2017.

[7] Y. Song, J. Crowcroft, and J. Zhang, "Automatic epileptic seizure detection in eegs based on optimized sample entropy and extreme learning machine," *Journal of Neuroscience Methods*, vol. 210, no. 2, pp. 132 – 146, 2012.

[8] B. He, D. Xu, R. Nian, M. van Heeswijk, Q. Yu, Y. Miche, and A. Lendasse, "Fast face recognition via sparse coding and extreme learning machine," *Cognitive Computation*, vol. 6, no. 2, pp. 264–277, 2014, cited By 38.

[9] Y. Zeng, X. Xu, D. Shen, Y. Fang, and Z. Xiao, "Traffic sign recognition using kernel extreme learning machines with deep perceptual features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1647–1653, 2017, cited By 45.

[10] Q. Lv, X. Niu, Y. Dou, J. Xu, and Y. Lei, "Classification of hyperspectral remote sensing image using hierarchical local-receptive-field-based extreme learning machine," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 3, pp. 434–438, 2016.

[11] Y. Zhou and Y. Wei, "Learning hierarchical spectral-spatial features for hyperspectral image classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 7, pp. 1667–1678, 2016.

[12] Y. Cai, X. Liu, and Z. Cai, "Bs-nets: An end-to-end framework for band selection of hyperspectral image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 3, pp. 1969–1984, 2020.

[13] Y. Xu, Z. Y. Dong, J. H. Zhao, P. Zhang, and K. P. Wong, "A reliable intelligent system for real-time dynamic security assessment of power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1253–1263, Aug 2012.

[14] X. Chen, Z. Y. Dong, K. Meng, Y. Xu, K. P. Wong, and H. W. Ngan, "Electricity price forecasting with extreme learning machine and bootstrapping," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 2055–2062, Nov 2012.

[15] Z. Wang, T. Chen, J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, pp. 1021–1028.

[16] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2224–2232.

[17] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and

M. Sun, "Graph neural networks: A review of methods and applications," *CoRR*, vol. abs/1812.08434, 2018. [Online]. Available: http://arxiv.org/abs/1812.08434

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[19] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, Dec 2014.

[23] Y. Zhou, J. Peng, and C. L. P. Chen, "Extreme learning machine with composite kernels for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2351–2360, 2015.

[24] Y. Cai, X. Liu, Y. Wu, P. Hu, R. Wang, B. Wu, and Z. Cai, *Extreme Learning Machine Based onÂ Evolutionary Multi-objective Optimization*. Singapore: Springer Singapore, 2017, pp. 420–435.

[25] A. Majumdar, "Graph structured autoencoder," *Neural Networks*, vol. 106, pp. 271 – 280, 2018.

[26] Pan Ji, M. Salzmann, and Hongdong Li, "Efficient dense subspace clustering," in *IEEE Winter Conference on Applications of Computer Vision*, March 2014, pp. 461–468.

[27] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, Nov 2013.

[28] C. Chen, W. Li, H. Su, and K. Liu, "Spectral-spatial classification of hyperspectral image based on kernel extreme learning machine," *Remote Sensing*, vol. 6, no. 6, p. 5795, 2014.

[29] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, vol. 99, 1999, pp. 200–209.

[30] Y. Li, C. Guan, H. Li, and Z. Chin, "A self-training semi-supervised svm algorithm and its application in an eeg-based brain computer interface speller system," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1285 – 1294, 2008.

[31] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of machine learning research*, vol. 7, no. Nov, pp. 2399–2434, 2006.