Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 412957, 11 pages
http://dx.doi.org/10.1155/2015/412957

Research Article

One-Class Classification with Extreme Learning Machine

Qian Leng,[1] Honggang Qi,[1] Jun Miao,[2] Wentao Zhu,[2] and Guiping Su[1]

[1]*School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 101408, China*
[2]*Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China*

Correspondence should be addressed to Jun Miao; jmiao@ict.ac.cn

Received 13 August 2014; Revised 8 November 2014; Accepted 10 November 2014

Academic Editor: Zhan-li Sun

One-class classification problem has been investigated thoroughly for past decades. Among one of the most effective neural network approaches for one-class classification, autoencoder has been successfully applied for many applications. However, this classifier relies on traditional learning algorithms such as backpropagation to train the network, which is quite time-consuming. To tackle the slow learning speed in autoencoder neural network, we propose a simple and efficient one-class classifier based on extreme learning machine (ELM). The essence of ELM is that the hidden layer need not be tuned and the output weights can be analytically determined, which leads to much faster learning speed. The experimental evaluation conducted on several real-world benchmarks shows that the ELM based one-class classifier can learn hundreds of times faster than autoencoder and it is competitive over a variety of one-class classification methods.

1. Introduction

One-class classification [1, 2] has received much interest during recent years, which has also been known as novelty or outlier detection. Different from normal classification, data samples from only one class, called the target class, are well characterized, while there are no or few samples from the other class (also called the outlier class). To reveal the necessity of one-class classification, we take the case of online shopping service as an example. In order to recommend goods users want, it is convenient to track the users' history shopping lists (positive training samples), while collection of negative training samples is challenging because it is hard to say which one users dislike. Other applications include machine fault detection [3], disease detection [4], and credit scoring [5]. The goal is to "teach" the classifier through observing target samples so that it can be applied to select unknown samples similar to the target class and reject samples which deviate significantly from the target class.

Various types of one-class classifier have been designed and applied in different fields; see [6] for a comprehensive review. Early attempt to obtain a one-class classifier is by estimating the probability density functions based on training data. Parzen density estimation [7, 8] superposes kernel functions on individual training samples to estimate the probability density function. Naive Parzen density estimation, similar to Naive Bayes approach used for classification, fits a Parzen density estimation on each individual feature and multiplies the results for final density estimation. A test sample is rejected if its estimated probability is below a threshold. However, estimating the true density distribution usually requires a large number of training samples.

A simpler task is to find the domain of the data distribution. Schölkopf et al. [9] constructed a hyperplane which is maximally distant from the origin to separate the regions that contain no data. An alternative approach is to find a hypersphere [10] instead of a hyperplane to include the most target data with the minimum radius. Both approaches are cast out in the form of quadratic programming, while some approaches [11–13] are of linear programming. One-class LP classifier [11] minimizes the volume of the prism, which is cut by a hyperplane that bounds the data from above with some mild constrains on dissimilarity representations. Lanckriet et al. [13] propose the one-class minimax probability machine that minimizes the worst case probability of misclassification of test data, using only the mean and covariance matrix

of the target distribution. When kernel methods are used, the aforementioned domain-based classifiers [2] can obtain more flexible descriptions. Recently, a minimum spanning tree based one-class classifier [14] was proposed. It considered graph edges as additional set of virtual target objects. By constructing a minimum spanning tree, recognition of a new test sample is determined by the shortest distance to the closest edge of that tree.

Autoencoder neural network is one of the reconstruction methods [1] to build a one-class classifier. The simplest architecture of such model is based on the single-hidden layer feed-forward neural networks (SLFNs). Usually, the hidden layer contains a smaller number of nodes than the number of input nodes which works like an information bottleneck. The classifier reproduces the input patterns at the output layer through minimizing the reconstruction error. However, standard backpropagation (BP) algorithm is used to train the networks, which is quite time-consuming. Extreme learning machine [15, 16] is originally developed to address the slow learning speed problem of gradient based learning algorithms for its iterative tuning of the networks' parameters. It randomly selects all parameters of the hidden neurons and analytically determines the output weights. It is stated [17, 18] in theory that ELM tends to provide the best generalization performance at extreme learning speed since it is a simple tuning-free algorithm.

In this paper, the proposed one-class classifier based on ELM is constructed for situations where only the target class is well described. The proposed one-class classifier utilizes the unified ELM learning theory [17], which leads to extreme learning speed and superior generalization performance. Moreover, the classifier further lessens the human intervention since it is not limited to specific target labels. Both random feature mappings and kernels can be adopted for such classifier which makes it more flexible to unique target descriptions. Constructing the proposed classifier for three quite different specific-designed artificial datasets demonstrates the classifier's ability to describe universal target class distributions. When real-world datasets are evaluated, the proposed one-classifier is competitive over a variety of one-class models and learns hundreds of times faster than autoencoder neural network for one-class classification.

The rest of the paper is organized as follows. Section 2 briefly reviews extreme learning machine. In Section 3, we first describe the hypersphere perceptron as a one-class classifier and then introduce our proposed ELM based one-class classifier. Section 4 describes the experiments conducted on both artificial and real-world datasets. Finally, Section 5 presents the conclusion of the work.

2. Brief Review of ELM

ELM aims to reach not only the smallest training error but also the smallest norm of output weights [16] between the hidden layer and the output layer. According to Bartlett's theory [19], the smaller norm of weights is, the better generalization performance of networks tends to have. Thus, better generalization performance can be expected for ELM networks. In [17], equality constraints are used in ELM, which provides a unified solution for regression, binary, and multiclass classifications.

2.1. Equality-Optimization-Constraints-Based ELM. Given N training data $(\mathbf{x}_i, t_i)_{i=1}^N$, where $\mathbf{x}_i = [x_{i1}, \ldots, x_{in}]^T \in R^n$ is the individual feature vector with dimension n and $t_i \in R^m$ is the desired target output, in the one-class classification case, single output node ($m = 1$) is enough. The ELM output function can be formulated as

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \sum_{j=1}^{L} \beta_j G\left(\mathbf{w}_j, b_j, \mathbf{x}\right), \tag{1}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^T$ is the vector of the output weights between the hidden layer and the output layer, $\mathbf{w}_j = [w_{j1}, \ldots, w_{jn}]^T$ is the input weights connecting input nodes with the jth hidden node, b_j is the bias of the jth hidden node, $\mathbf{h}(\mathbf{x}) = [G(\mathbf{w}_1, b_l, \mathbf{x}), \ldots, G(\mathbf{w}_L, b_L, \mathbf{x})]^T$ is the output vector of the hidden layer with respect to input $\mathbf{x}$, and $G(\mathbf{w}, b, \mathbf{x})$ is the activation function (e.g., sigmoid function $G(\mathbf{w}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{w}^T \cdot \mathbf{x} + b)))$) satisfying ELM universal approximation capability theorems [20, 21]. In fact, $\mathbf{h}(\mathbf{x})$ is a known nonlinear feature mapping which maps the training data $\mathbf{x}$ from the n-dimensional input space to the L-dimensional ELM feature space [17]. The goal of ELM is to minimize the norm of output weights as well as the training errors, which is equivalent to

$$\begin{aligned} \min \quad & L_{P_{\mathrm{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \\ \text{s.t.} \quad & \mathbf{h}\left(\mathbf{x}_i\right)^T \boldsymbol{\beta} = t_i - \xi_i, \quad i = 1, \ldots, N, \end{aligned} \tag{2}$$

where ξ_i is the slack variable of the training sample $\mathbf{x}_i$ and C controls the tradeoff between the output weights and the errors. Based on the Karush-Kuhn-Tucker (KKT) theorem [22], the corresponding Lagrange function of the primal ELM optimization (2) is

$$L_{D_{\mathrm{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\alpha_i\left(\mathbf{h}\left(\mathbf{x}_i\right)^T\boldsymbol{\beta} - t_i + \xi_i\right); \tag{3}$$

the following optimality conditions of (3) should be satisfied:

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \boldsymbol{\beta}} = 0 \Longrightarrow \boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i, \quad \mathbf{h}\left(\mathbf{x}_i\right) = \mathbf{H}^T\boldsymbol{\alpha}, \tag{4a}$$

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \xi_i} = 0 \Longrightarrow \alpha_i = C\xi_i, \quad i = 1, \ldots, N, \tag{4b}$$

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \alpha_i} = 0 \Longrightarrow \mathbf{h}\left(\mathbf{x}_i\right)^T\boldsymbol{\beta} - t_i + \xi_i = 0, \quad i = 1, \ldots, N, \tag{4c}$$

where $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_N)]^T$ is the hidden layer output matrix and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ is the vector of Lagrange variables. Substituting (4a) and (4b) into (4c) we have

$$\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)\boldsymbol{\alpha} = \mathbf{T}. \tag{5}$$

Here $\mathbf{I}$ is the identity matrix and $\mathbf{T} = [t_1, \ldots, t_N]^T$. Substituting (5) into (4a), we get

$$\boldsymbol{\beta} = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \tag{6}$$

The ELM output function (1) can be further derived as

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \mathbf{h}(\mathbf{x})^T \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \tag{7}$$

If the hidden nodes' feature mapping $\mathbf{h}(\mathbf{x})$ is unknown to users, kernel methods that satisfy Mercer's condition can be adopted: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j)$. The ELM kernel output function can be written as

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \mathbf{h}(\mathbf{x})^T \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\text{ELM}} \right)^{-1} \mathbf{T} \end{aligned} \tag{8a}$$

and the kernel matrix for ELM is

$$\boldsymbol{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega_{\text{ELM}i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \tag{8b}$$

2.2. Advances of ELM. Extreme learning machine has gained much more popularity since its advent. It has been able to avoid the problem of time complexity which classic learning techniques are confronted with while providing better generalization performance with less human intervention. Because of such attractive features, researchers have extended the basic ELM to several different directions and many variants of ELM have been developed. For instance, online sequential ELM (OS-ELM) [23, 24] can learn the sequential coming data (one by one or chunk by chunk) with a small effort to update the output weights. The training data are discarded after being learned by the network and the output weights need not be retrained, which is especially efficient for time-series problems. Other typical works include fully complex ELM [25, 26], incremental ELM (I-ELM) [20, 21], sparse ELM [27], ELM with elastic output [28, 29], and ELM ensembles [30–32]. See [33] for further details on the many ELM variants.

When uncertainty is present in the dataset, integration of fuzzy logic system and extreme learning machine tends to enhance the generalization capability of ELM. In [34], a neurofuzzy Takagi-Sugeno-Kang (TSK) fuzzy inference system is constructed utilizing extreme learning machine. The number of inference rules is previously determined by the k-means method. One ELM is used to obtain the membership of each fuzzy rule and multiple ELM are used to obtain the consequent part. Rong et al. [35] show that type-1 fuzzy inference system (type-1 FLS) is equivalent to a generalized SLFN. Hence, the hidden nodes work as the antecedent part and the output weights as the consequent part. Then, extreme learning machine is directly applied to the type-1 FLS and the corresponding online sequential fuzzy ELM has also been developed. Deng et al. [36] further extend the idea to type-2 fuzzy inference system (type-2 FLS) because of type-2 FLS's superiority in modeling high level uncertainty. With the most widely used interval type-2 FLS, the parameters of the antecedents are randomly initialized according to the ELM mechanism. The Moore-Penrose generalized inverse is used to initialize the parameters of the consequents and the parameters are finally refined by Karnik-Mendel algorithm [37]. Many applications have also been investigated in the literature. For example, the hybrid model of ELM with interval type-2 FLS has been applied for permeability prediction [38].

3. The Proposed One-Class Classifier

3.1. Support Vector Data Description. For a better understanding of one-class classifiers, support vector data description (SVDD) [10] is discussed here for one-class classification process. SVDD defines a spherically shaped boundary around the complete target set and is intuitively appealing since it regards the target class as a self-closed system. Let $X = \{\mathbf{x}_i, i = 1, \ldots, N\}$ be the training set, and $\mathbf{x}_i \in R^n$ is drawn from the target distribution. SVDD aims to minimize the volume of the sphere as well as the training errors ξ_i for objects falling outside the boundary, which is equivalent to

$$\begin{aligned} \min \quad & L_{P_{\text{SVDD}}} = R^2 + C \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \left\| \mathbf{x}_i - \mathbf{a} \right\|^2 \le R^2 + \xi_i, \quad i = 1, \ldots, N \\ & \xi_i \ge 0, \quad \forall i, \end{aligned} \tag{9}$$

where R and $\mathbf{a}$ are the hypersphere's radius and center, respectively. Parameter C controls the tradeoff between the volume and the errors.

The corresponding function of the primal SVDD optimization (9) is

$$\begin{aligned} & L_{\text{SVDD}} \\ & \quad = R^2 + C \sum_{i=1}^{N} \xi_i \\ & \qquad - \sum_{i=1}^{N} \alpha_i \left(R^2 + \xi_i - \left(\left\| \mathbf{x}_i \right\|^2 - 2\mathbf{a} \cdot \mathbf{x}_i + \left\| \mathbf{a} \right\|^2 \right) \right) - \sum_{i=1}^{N} \beta_i \xi_i \end{aligned} \tag{10}$$

with the Lagrange variables $\alpha_i \ge 0$ and $\beta_i \ge 0$. L_{SVDD} should be minimized with respect to R, $\mathbf{a}$, ξ_i and maximized with respect to α_i, β_i.

Based on the Karush-Kuhn-Tucker (KKT) theorem [22], to get the optimal solutions of (10), we should have

$$\frac{\partial L_{\text{SVDD}}}{R} = 0 \Longrightarrow \sum_{i=1}^{N} \alpha_i = 1, \tag{11a}$$

$$\frac{\partial L_{\text{SVDD}}}{\mathbf{a}} = 0 \Longrightarrow \mathbf{a} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i, \tag{11b}$$

$$\frac{\partial L_{\text{SVDD}}}{\xi_i} = 0 \Longrightarrow C = \alpha_i + \beta_i. \tag{11c}$$

From (11c) and $\alpha_i \geq 0$ and $\beta_i \geq 0$, β_i can be removed and α_i can be further limited to the interval $[0, C]$:

$$0 \leq \alpha_i \leq C. \tag{12}$$

Substituting (11a)–(11c) into (10), the dual optimization function can be derived as

$$L_{D_{\text{SVDD}}} = \sum_{i=1}^{N} \alpha_i \left(\mathbf{x}_i \cdot \mathbf{x}_i\right) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{13}$$

subject to constraints (11a) and (12). To constitute a flexible data description model, kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j)$, with an implicit feature mapping $\boldsymbol{\phi}$ of the data into a higher dimensional feature space, can be adopted to replace the inner product $(\mathbf{x}_i \cdot \mathbf{x}_j)$. In this case, the corresponding dual optimization function is changed to

$$L_{D_{\text{SVDD}}} = \sum_{i=1}^{N} \alpha_i K\left(\mathbf{x}_i, \mathbf{x}_i\right) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K\left(\mathbf{x}_i, \mathbf{x}_j\right). \tag{14}$$

The KKT conditions of the target functions are

$$\begin{gathered} \alpha_i \left(R^2 + \xi_i - \left\|\mathbf{x}_i - \mathbf{a}\right\|^2\right) = 0, \\ \beta_i \xi_i = 0. \end{gathered} \tag{15}$$

The constraints have to be enforced and we have three cases as follows:

(1) $\alpha_i = 0$

$$\begin{gathered} \beta_i = C \Longrightarrow \xi_i = 0, \\ \alpha_i = 0 \Longrightarrow R^2 \geq \left\|\mathbf{x}_i - \mathbf{a}\right\|^2, \end{gathered} \tag{16}$$

(2) $0 < \alpha_i < C$

$$\begin{gathered} \beta_i > 0 \Longrightarrow \xi_i = 0, \\ 0 < \alpha_i < C \Longrightarrow R^2 = \left\|\mathbf{x}_i - \mathbf{a}\right\|^2, \end{gathered} \tag{17}$$

(3) $\alpha_i = C$

$$\begin{gathered} \beta_i = 0 \Longrightarrow \xi_i \geq 0, \\ \alpha_i = C \Longrightarrow R^2 \leq \left\|\mathbf{x}_i - \mathbf{a}\right\|^2. \end{gathered} \tag{18}$$

Only a small ratio of objects with $\alpha_i > 0$ are called the support vectors. The dual optimization functions (13) and (14) are standard Quadratic Programming (QP) problems and the Lagrange variables α_i can be obtained using some optimization methods such as SMO algorithm [39]. To test a new object $\mathbf{z}$, its distance to the center of the sphere is calculated. The classifier will accept the object if the distance is less than or equal to the radius:

$$\begin{aligned} \left\|\mathbf{z} - \mathbf{a}\right\|^2 &= (\mathbf{z} \cdot \mathbf{z}) \\ &\quad - 2\sum_{i=1}^{N} \alpha_i \left(\mathbf{z} \cdot \mathbf{x}_i\right) + \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \leq R^2. \end{aligned} \tag{19}$$

In addition to the batch learning model of SVDD, incremental learning methods [40] of SVM are extended to SVDD algorithm. Yin et al. [28] show an online fault diagnosis process through a hybrid model of incremental SVDD (ISVDD) and ELM with incremental output structure (IOELM). They used the ISVDD to detect the unknown failure model, and the output nodes of IOELM are adaptively increased to recognize the new failure mode.

3.2. The ELM Based One-Class Classifier. When data only from the target class are available, the one-class classifier is trained to accept target objects and reject objects that deviate significantly from the target class. In the training phase, the one-class classifier, which defines a distance function d between the objects and the target class, takes in the training set X to build the classification model. In general, the classification model contains two important parameters to be determined: threshold θ and modal parameter λ. A generic test sample $\mathbf{z}$ is accepted by the classifier if $d(\mathbf{z} \mid X, \lambda) < \theta$.

In the training phase, not all the training samples are to be accepted by the one-class classifier due to the presence of outliers or noisy data contained in the training set. Otherwise, the trained classification model may generalize poor to unknown test set when the training set includes abnormal data samples. Usually, threshold θ is determined such that a user-specified fraction μ of training samples most deviant from the target class are rejected. For instance, if one is told five percent of training samples are mislabeled, setting $\mu = 0.05$ makes the classifier more robust. Even when all the samples are correctly labeled, rejecting a small fraction of training samples helps the classifier to learn the most representative model from the training samples.

Any one-class classifier has model parameters which influence the model complexity (flexibility), for example, the number of hidden nodes in autoencoder neural networks or the tradeoff parameter C of SVDD. Minimizing the errors of both the target and outlier classes on a cross-validation set is no longer available since there is no data from the outlier class. Fortunately, several model selection criteria [2] have been proposed. Assuming the uniform distribution of the outlier class, consistency-based model selection [41] method is one of the most effective methods used to select the model parameters. The basic idea is that the complexity of the classifier can be increased as long as it still fits the target data. The more complex the model, the smaller the volume of the classifier in the object space and the less the probability of outlier objects falling inside the domain of the classifier. In practice, one can make an ordering of the potential model parameters such that the latter parameter

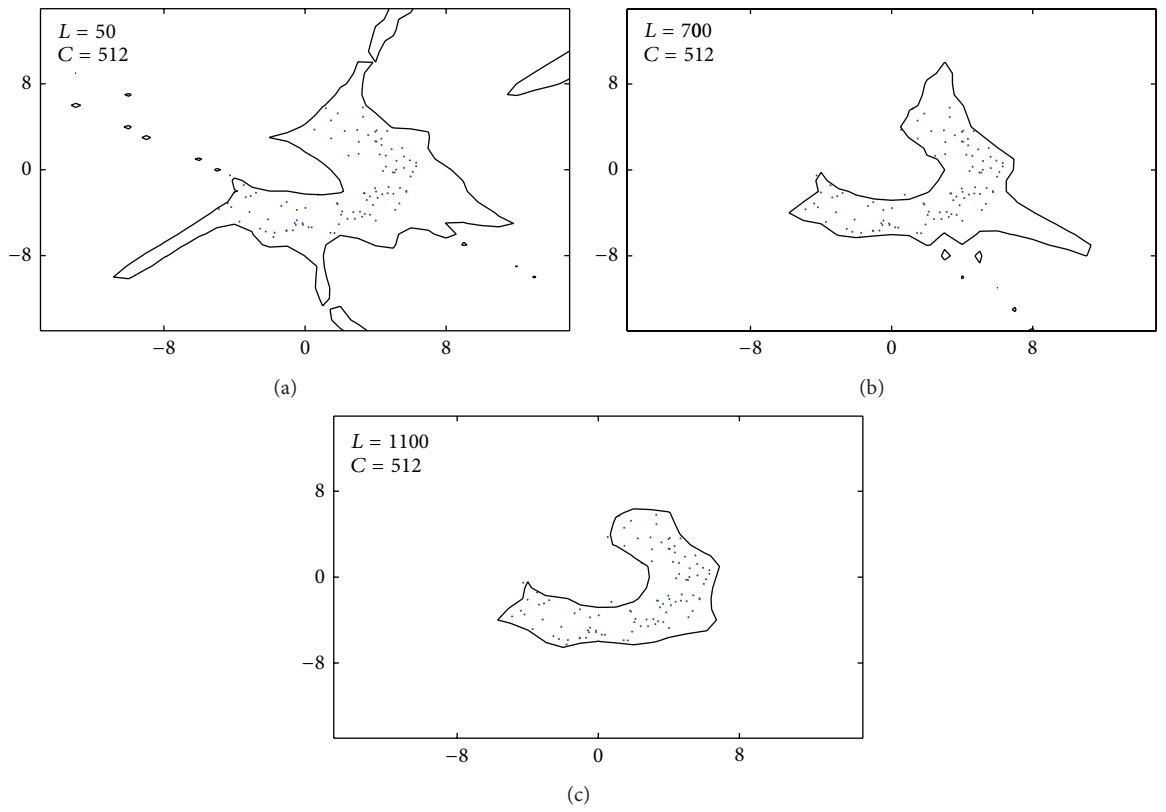


Figure 1: Example boundaries of the proposed classifier with different number of hidden nodes.

always yields the more complex classifier and chooses the most complex classifier without overfitting the target data.

The compactness hypothesis [42] is the basis for object recognition. It states that similar real world objects have to be close in the feature space. Therefore, for similar objects from the target class, the target outputs should be the same:

$$t_i = y, \quad \forall \mathbf{x}_i \in X, \tag{20}$$

where y is a real number. All the training samples' target outputs are set to the same value y. Then, the desired target output vector is $\mathbf{T} = [t_1, \ldots, t_N]^T = [y, \ldots, y]^T$. Training the samples from the target class can directly use the optimization function (2). For a new test sample $\mathbf{z}$, the distance function between the sample object and the target class is defined as

$$\begin{aligned} d_{\mathrm{ELM}}(\mathbf{z} \mid X, \lambda) &= \left| \mathbf{h}(\mathbf{z})^T \boldsymbol{\beta} - y \right| \\ &= \left| \mathbf{h}(\mathbf{z})^T \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} - y \right|. \end{aligned} \tag{21}$$

The decision whether $\mathbf{z}$ belongs to the target class or not is based on threshold θ. Recall that θ is optimized to reject a small fraction μ of training samples to avoid overfitting. The distances of the training samples to the target class can be directly determined using (21) and the constraint of (2)

$$d_{\mathrm{ELM}}(\mathbf{x}_i \mid X, \lambda) = \left| \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} - y \right| = |\xi_i|. \tag{22}$$

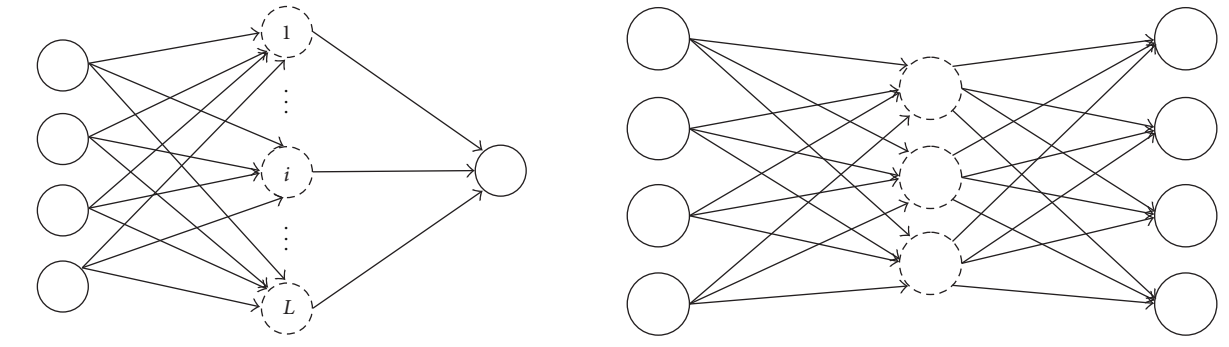
From (22), we find the distances are $|\xi_i|$ and the larger $|\xi_i|$ means the more deviant of the training sample $\mathbf{x}_i$ from the target class. Hence, we derive threshold θ based on a quantile function to reject the most deviant training samples. Denote the sorted sequence of the distances of training samples by $\mathbf{d} = [d_{(1)}, \ldots, d_{(N)}]$ such that $d_{(1)} \geq \cdots \geq d_{(N)}$. Here, $d_{(1)}$ and $d_{(N)}$ represent the most and the least deviant samples. The function determining θ can be written as

$$\theta = d_{\mathrm{floor}(\mu \cdot N)}, \tag{23}$$

where floor(a) returns the largest integer not greater than a. Then, we can get the decision function for $\mathbf{z}$ to the target class:

$$\begin{aligned} \mathscr{C}_{\mathrm{ELM}}(\mathbf{z}) &= \mathrm{sign}\left(\theta - d_{\mathrm{ELM}}(\mathbf{z} \mid X, \lambda)\right) \\ &= \begin{cases} 1 & z \text{ is classified as a target} \\ -1 & z \text{ is classified as a outlier.} \end{cases} \end{aligned} \tag{24}$$

Remark 1. The target output y can be assigned to arbitrary real number except 0. When $y = 0$, seen from (6), the output weights between the hidden layer and the output layer become 0 ($\boldsymbol{\beta} = \mathbf{0}$, $\mathbf{0}$ is the L-dimensional zero vector). Therefore, the decision value of any sample $\mathbf{z}$ is 0 using the proposed classifier. It is obvious that, in such case, the one-class classifier cannot distinguish between the target class and the outlier class. When $y \neq 0$, as there are infinite possible y, there seem to exist infinite ELM based one-class classifiers. To



(a) ELM network (b) Autoencoder neural network

Figure 2: Comparisons between ELM network and autoencoder neural network: the number of hidden nodes in ELM network should be large enough according to ELM learning theory and one output node is enough for one-class classification, while the number of hidden nodes in autoencoder neural network is usually less than the feature dimension and the number of output nodes should be equal to the number of input nodes.

get a universal ELM based one-class classifier, we normalize the distance function (21) by dividing the target output y

$$\begin{aligned} & d_{\text{NORM-ELM}}(\mathbf{z} \mid X, \lambda) \\ &= \left| \frac{\mathbf{h}(\mathbf{z})\boldsymbol{\beta} - y}{y} \right| = \left| \frac{\left(\mathbf{h}(\mathbf{z})\mathbf{H}^T \left(\mathbf{I}/C + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} - y\right)}{y} \right| \\ &= \left| \frac{\mathbf{h}(\mathbf{z})\mathbf{H}^T \left(\mathbf{I}/C + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{e}y - y}{y} \right| \\ &= \left| \mathbf{h}(\mathbf{z})^T \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{e} - 1 \right|, \end{aligned} \tag{25}$$

where $\mathbf{e}$ is the N-dimensional unit vector. The normalization formula (25) is to eliminate the possible bias introduced by the target output y. In practice, one can set the target output $y = 1$ such that (21) is equivalent to (25) and the normalization step is implicitly done.

Remark 2. Both random feature mappings and kernels can be used for the proposed one-class classifier. When nonlinear piecewise continuous functions satisfying ELM universal approximation capability theorems [20, 21] are used as the activation function, the ELM network can approximate any target continuous function as long as the number of hidden nodes L is large enough. When the feature mapping is unknown, kernel methods can be adopted as shown in (8a) and (8b). Huang et al. [17] have shown ELM, the unified solution for regression, binary, and multiclass classifications. Since the same optimization formula (2) is used in the proposed one-class classifier, this paper also shows ELM, the unified learning mode for one-class classification.

Figure 1 shows the decision boundaries (black curves) of the classifier with incremental hidden nodes using sigmoid function as the activation function. The dataset (blue points) is composed of 100 samples in the plane. Threshold θ is determined such that $\mu = 0.01$ and the model parameter C is automatically determined by the consistency-based model selection method. When the number of hidden nodes is small ($L = 50$), the classifier fails to approximate the target region and some unexpected "holes" without any targets can be seen from the leftmost picture of Figure 1. The weakness alleviates as more hidden nodes are added. When the number of hidden nodes L gets large enough, the classifier can be close enough to describe the target class well. This is consistent with ELM universal approximation capability theorems [20, 21].

Table 1: Specification of UCI datasets.

Datasets	Target class	# target	# outlier	# features
Spectf heart	0	55	212	44
Arrhythmia	Normal	245	207	279
Sonar	Mines	111	97	60
Liver	Healthy	145	200	6
E. coli	Periplasm	52	284	7
Diabetes	Present	500	268	8
Breast	Benign	241	458	9
Abalone	Classes 1–8	1407	2770	8

Remark 3. Autoencoder is one of the most effective neural networks approaches for one-class classification, which has been applied by Manevitz and Yousef for document retrieval [43]. Constrain the number of output nodes that must be equal to the number of input nodes ($m = n$). The hidden layer in such a network actually acts as a bottleneck, where $L < n$. The idea is that while the bottleneck prevents learning the full identity function on n-space, the identity on the small set of examples is in fact learnable. Traditional learning algorithms like BP are used to train the network. Several challenging issues such as local minimum, trivial human intervention,

Table 2: The value of F_1 measure with standard deviations (in parentheses) for a number of one-class classifiers. Twenty trials have been conducted for each dataset.

Dataset	SPECTF heart	Arrhythmia	Sonar	Liver
Classifier		F_1		
Naive Parzen	41.7 (4.2)	61.8 (1.1)	46.8 (2.2)	41.5 (0.9)
Parzen	39.3 (1.7)	63.7 (1.2)	49.8 (2.9)	40.7 (1.4)
k-means	38.3 (4.7)	63.7 (1.7)	53.2 (3.2)	41.7 (1.4)
1-NN	31.8 (2.6)	59.2 (1.5)	**60.4 (2.2)**	41.3 (1.3)
k-NN	34.7 (1.2)	62.4 (0.9)	55.3 (1.3)	42.0 (1.2)
Autoencoder	39.3 (3.4)	**64.8 (1.6)**	50.6 (2.4)	42.2 (1.7)
PCA	NaN[1]	26.3 (5.3)	37.2 (8.3)	41.1 (1.3)
MST	33.7 (1.7)	62.4 (0.8)	56.7 (1.8)	42.1 (1.1)
k-centers	36.4 (2.9)	62.8 (1.2)	53.3 (2.3)	41.6 (1.3)
SVDD	38.9 (4.7)	60.5 (4.8)	51.2 (5.8)	40.6 (3.1)
MPM	31.1 (8.7)	51.9 (5.0)	44.6 (6.3)	40.7 (2.0)
LPDD	38.3 (3.9)	63.8 (2.0)	52.2 (4.2)	40.7 (1.6)
SVM	38.1 (6.4)	63.4 (1.9)	53.6 (3.1)	40.5 (2.4)
ELM	**42.6 (1.8)**	63.6 (1.6)	54.2 (3.5)	**43.0 (1.6)**

[1]None of target data is recalled.

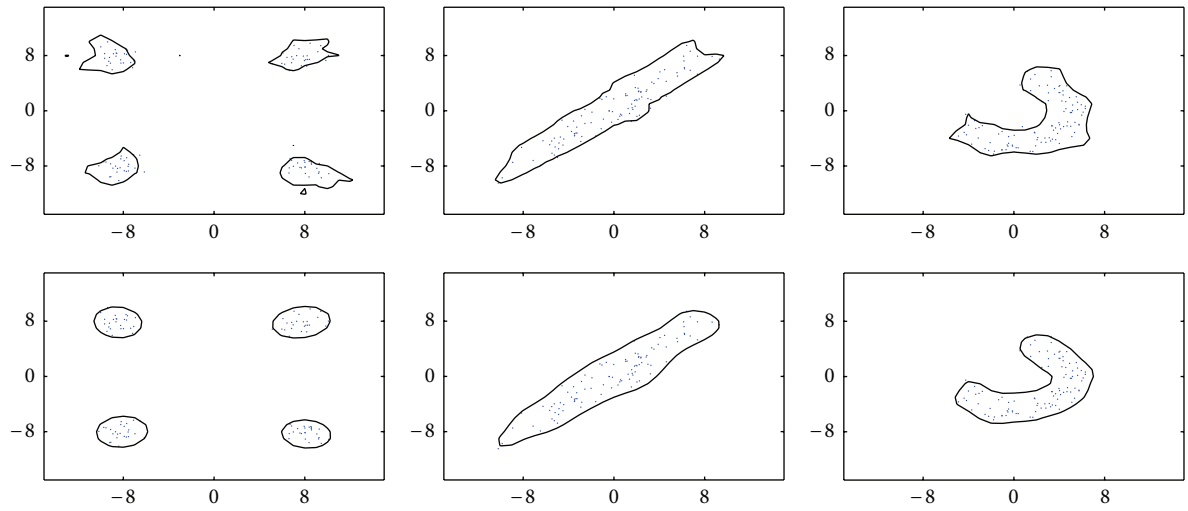
Table 3: The value of F_1 measure with standard deviations (in parentheses) for a number of one-class classifiers. Twenty trials have been conducted for each dataset.

Dataset	*E. coli*	Diabetes	Breast	Abalone
Classifier		F_1		
Naive Parzen	71.7 (7.0)	68.7 (1.1)	82.4 (3.2)	51.8 (0.3)
Parzen	75.1 (5.7)	67.7 (1.4)	80.1 (7.7)	49.0 (1.0)
k-means	54.6 (15.2)	68.9 (1.1)	58.8 (17.2)	45.2 (1.3)
1-NN	21.2 (3.9)	64.8 (0.9)	35.3 (5.8)	35.7 (1.0)
k-NN	43.9 (14.2)	68.8 (1.0)	34.9 (7.5)	49.8 (1.4)
Autoencoder	53.5 (15.8)	66.9 (1.3)	37.9 (10.9)	48.7 (2.7)
PCA	33.7 (15.4)	65.5 (1.9)	31.1 (1.0)	46.0 (0.5)
MST	36.3 (12.9)	67.5 (0.7)	34.4 (3.4)	47.3 (0.9)
k-centers	38.8 (6.5)	67.7 (1.1)	49.4 (22.9)	42.5 (2.6)
SVDD	50.9 (10.6)	61.1 (2.5)	68.0 (9.1)	44.5 (3.3)
MPM	38.8 (11.8)	63.5 (1.7)	71.7 (6.5)	44.9 (1.9)
LPDD	67.8 (11.0)	66.6 (0.8)	79.6 (7.5)	45.8 (2.0)
SVM	57.2 (12.8)	66.6 (1.1)	**83.1 (3.0)**	46.2 (1.2)
ELM	**77.1 (4.8)**	**69.1 (1.1)**	80.1 (5.1)	**53.0 (0.7)**

and time consuming in learning stage discourage people who are not familiar in the field to use it, while the ELM based one-class classifier can approximate the target class well as long as the dimensionality of the feature mappings is large enough (cf. Figure 2).

4. Experiments

4.1. Artificial Datasets. First, we illustrate the proposed method with both random feature mappings and kernels on three specific designed artificial datasets, which all contain 100 samples created in a 2D feature space. The first dataset contains four Gaussian distributions (each has 25 samples) with the same unit covariance matrix but with different mean vectors. It is set to test the classifier's sensitivity to multimodality. The second dataset contains one Gaussian distribution with the first feature with a variance of 1 and the second feature with a variance of 40. Moreover, the two features are rotated over 45 degrees to construct a strong correlation. The third banana-shaped dataset, which has been shown in Section 3, contains one uniform distribution along an arc curve with some small position offsets. It is to test the influence of convexity. In Figure 3, the datasets (blue points) together with the decision boundaries (black curves) in the feature space are illustrated. Sigmoid function acts as



(a) Multimodal dataset (b) Scale and rotate dataset (c) Banana dataset

Figure 3: The upper row shows the boundaries of the method with random feature mappings and the bottom row shows the boundaries of the method with Gaussian kernel. Parameter C of the method with random feature mapping from left to right is 2^{11}, 2^{6}, and 2^{9}. Parameters (C, σ) of the method with Gaussian kernel from left to right are $(2^{2}, 5.76)$, $(2^{0}, 2.81)$, and $(2^{-4}, 1.55)$.

the activation function for the method with random feature mappings (L large enough) and Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2)$ is used for the method with kernels. All the thresholds are determined such that $\mu = 0.01$. The pictures show that methods using both random feature mappings and kernels give reasonable results. However, the method with kernels tends to be superior to the method with random feature mappings since the boundary captures the distribution more precisely, while in Figure 3(a) some small "holes" still exist in the upper left and lower right regions for the method with random feature mappings.

4.2. UCI Datasets. This section compares the performance of the proposed method with a variety of one-class classification algorithms. The popular one-class classifiers to be compared include Parzen [7], Naive Parzen, k-means [44], k-centers [45], 1-NN [46], k-NN [47], autoencoder, PCA [48], MST [14], MPM [13], SVDD [10], LPDD [11], and SVM [9]. The implementations for one-class SVM are carried out using compiled C-coded SVM packages: LIBSVM [49]. All the other algorithms are conducted with Matlab toolbox DD_TOOLS [50]. Binary and multiclass classification datasets taken from UCI Machine Learning Repository [51] are used. The specifications of the datasets are shown in Table 1. The datasets are transformed for one-class classification by setting a chosen class as the target class and all the other classes as the outlier class.

In our experiments, all the inputs have been normalized into range $[0, 1]$. The samples from the target class are equally partitioned in two sets for training and testing, respectively. All one-class classifiers are trained on target data only and tested on both the remaining target data and all other nontarget data. To assess the performance, we use F_1 measure [52], which is defined as a combination of recall (R) and precision (P) with an equal weight in the following form:

$$F_1(R, P) = \frac{2RP}{(R + P)}. \tag{26}$$

All the thresholds θ are determined such that $\mu = 0.1$. The Gaussian kernel is used in Parzen, Naive Parzen, MPM, SVDD, SVM, and ELM. The consistency-based model selection method is employed to select the model parameters. For

Table 4: The value of precision and recall for two datasets (arrhythmia and *E. coli*).

Dataset	Arrhythmia		*E. coli*	
Classifier	Precision	Recall	Precision	Recall
Naive Parzen	45.6	96.0	63.7	86.2
Parzen	52.0	82.4	71.5	80.6
k-means	52.5	81.0	52.4	64.6
1-NN	44.5	88.6	12.1	90.2
k-NN	47.8	90.2	31.9	87.3
Autoencoder	52.6	84.7	47.3	73.5
PCA	79.1	15.9	23.2	74.4
MST	47.3	91.8	23.9	90.0
k-centers	50.4	83.4	29.3	68.9
SVDD	55.7	69.3	48.8	66.9
MPM	64.0	43.9	38.4	45.4
LPDD	51.7	83.3	67.8	75.4
SVM	52.4	80.5	57.8	65.8
ELM	52.8	80.1	83.2	72.3

Table 5: Running time for ELM, autoencoder, and SVDD over twenty trials.

Classifier	ELM		Autoencoder		SVDD	
Dataset	Training time	Testing time	Training time	Testing time	Training time	Testing time
SPECTF heart	0.3	0.1	93.2	1.1	0.4	0.1
Arrhythmia	0.2	0.2	17462.0	8.7	1.6	0.2
Sonar	0.1	0.1	248.8	1.4	0.7	0.1
Liver	0.1	0.1	9.9	0.9	1.3	0.2
E. coli	0.1	0.1	8.2	0.9	0.7	0.2
Diabetes	0.3	0.2	37.3	0.9	6.2	0.2
Breast	0.1	0.2	45.3	0.9	2.5	0.3
Abalone	1.0	2.5	64.2	1.0	118.0	2.5

Parzen, MPM, SVDD, SVM, and ELM, the kernel parameter σ is chosen from 20 aliquots between the minimum and maximum pairwise object distances, so as the smoothing parameter of sigmoid transform function used in LPDD. For k-means, k-centers, parameter k is selected from the range $\{1, 2, \ldots, 20\}$. For ELM, another parameter C is chosen from the range $\{2^{-24}, 2^{-23}, \ldots, 2^{25}\}$ and we set σ a higher priority than C; that is, when the parameter combinations, (σ_1, C_1) and (σ_2, C_2), both obtain consistent boundaries, we always choose a smaller σ rather than a larger C. We try every possible parameter setting and find the most complex classifier as long as the classifier is consistent. For Naive Parzen and k-NN, the leave-one-out maximum likelihood estimation is used. One-class PCA retains 0.95 variance for the training set. For MST, the complete minimum spanning tree is used. The number of hidden nodes in autoencoder neural network is carefully chosen from a large range and the optimal number is selected.

All the experiments are carried out in Matlab R2013a environment running in E5504 2 GHz CPU, 4 GB RAM. Twenty trials have been conducted for each dataset and the average F_1 and corresponding standard deviations are shown in Tables 2 and 3. The best results are shown in boldface. As an example, we give a detailed description for the diabetes experiment. First, all the samples from both the target class and the outlier class are normalized into range $[0, 1]$. Then, the 500 training target samples are randomly divided into two equal sets (250 samples for each set). One of the sets is used for training the one-class classifier and the other set, together with all the samples from the outlier class, is used for testing only. After that, the consistency-based model selection method is employed to select the model parameters for each classifier using only the training set. Finally, the other target set with the outlier set is judged by the trained classifier with precision and recall recorded. F_1 value is then derived as (26). The same procedure repeats for twenty times and the corresponding mean and deviation values are calculated. It can be seen that the generalization performance of ELM is the best in five of the eight experiments while in the other experiments, except for sonar dataset, the performance is comparable to the best classifier. Table 4 presents a detailed performance comparison of two datasets, including precision and recall.

Table 5 reports the execution time comparisons in seconds between the ELM, autoencoder, and SVDD classifiers for all the eight experiments. As observed from Table 5, the advantage of the ELM on training time is quite obvious. ELM can generally learn hundreds of times faster than autoencoder neural network due to the tuning-free mechanism. Besides, ELM also learns much faster than SVDD without solving a QP problem. For testing time, since autoencoder may obtain a more compact network and the parameters have been tuned in the training phase, the computational time depends on the specific task. The computational complexity of ELM mostly depends on the number of samples while autoencoder depends on both the number of samples and the number of dimensions. Thus, for datasets with relatively small size and high dimensions, such as arrhythmia dataset, ELM obtains a smaller testing time, while for datasets with relatively large size and low dimensions, such as abalone dataset, autoencoder reacts faster to the testing samples. However, ELM still tends to outperform autoencoder with respect to both training time and accuracy. It is obvious that ELM and SVDD obtain a similar testing time since both of them utilize a kernel function.

5. Conclusion

This paper presents a simple and efficient one-class classifier utilizing extreme learning machine, which also shows ELM, the unified learning mode for one-class classification. Both random feature mappings and kernels can be used for the proposed classifier while the method with kernels tends to be superior to the method with random feature mappings. Moreover, the proposed classifier with kernels achieves the best results on five of the eight UCI datasets, which suggests ELM being effective for one-class classification problem. We have also discussed the relationships and differences between autoencoder neural network and ELM network for one-class classification. Although autoencoder neural network has been successfully applied in many applications, the slow gradient based method is still used to tune all the parameters, which is far slower than required. On the other hand, the ELM based one-class classifier has an analytical solution which can obtain superior generalization performance at much faster learning speed. Possible future directions include the fusion of fuzzy logic and ELM for one-class classification, one-class classifier ensembles with ELM, and substituting autoencoder with the ELM based one-class classifier for deep learning.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially sponsored by Natural Science Foundation of China (nos. 61175115, 61272320, 61379100, and 61472388). The authors would like to thank the helpful discussions with Mr. Fan Wang and Dr. Laiyun Qing.

References

[1] D. M. J. Tax, *One-class classification [Ph.D. thesis]*, Delft University of Technology, 2001.

[2] P. Juszczak, *Learning to recognise. A study on one-class classification and active learning [Ph.D. thesis]*, Delft University of Technology, Delft, Netherlands, 2006.

[3] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005.

[4] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, C. Pellegrini, and A. Geissbuhler, "An application of one-class support vector machines to nosocomial infection detection," in *Proceedings of Medical Informatics*, 2004.

[5] K. Kennedy, B. Mac Namee, and S. J. Delany, "Credit scoring: solving the low default portfolio problem using one-class classification," in *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 168–177, 2009.

[6] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[7] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[8] R. P. W. Duin, "On the choice of smoothing parameters for Parzen estimators of probability density functions," *IEEE Transactions on Computers*, vol. 25, no. 11, pp. 1175–1179, 1976.

[9] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[10] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[11] E. Pekalska, D. M. J. Tax, and R. P. W. Duin, "One-class LP classifier for dissimilarity representations," in *Neural Information Processing Systems*, pp. 761–768, MIT Press, Cambridge, Mass, USA, 2003.

[12] C. Campbell and K. P. Bennett, "A linear programming approach to novelty detection," in *Neural Information Processing Systems*, pp. 395–401, 2000.

[13] G. R. G. Lanckriet, L. El Ghaoui, and M. I. Jordan, "Robust novelty detection with single-class MPM," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and T. Obermayer, Eds., vol. 15, pp. 905–912, MIT Press, Cambridge, Mass, USA, 2003.

[14] P. Juszczak, D. M. J. Tax, E. Pękalska, and R. P. W. Duin, "Minimum spanning tree based one-class classifier," *Neurocomputing*, vol. 72, no. 7–9, pp. 1859–1869, 2009.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, pp. 985–990, Budapest, Hungary, July 2004.

[16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[17] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[18] W. Zhu, J. Miao, and L. Qing, "Constrained extreme learning machine: a novel highly discriminative random feedforward neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '14)*, Beijing, China, June 2014.

[19] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[20] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[21] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16-18, pp. 3056–3062, 2007.

[22] R. Fletcher, *Practical Methods of Optimization: Volume 2 Constrained Optimization*, Wiley, New York, NY, USA, 1981.

[23] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[24] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, vol. 34, no. 6, pp. 2284–2292, 2004.

[25] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, no. 1-4, pp. 306–314, 2005.

[26] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, no. 4–6, pp. 576–583, 2008.

[27] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.

[28] G. Yin, Y.-T. Zhang, Z.-N. Li, G.-Q. Ren, and H.-B. Fan, "Online fault diagnosis method based on incremental support vector data description and extreme learning machine with incremental output structure," *Neurocomputing*, vol. 128, pp. 224–231, 2014.

[29] T. Wang, S. Wang, and H. Zhang, "Dynamic extreme learning machine: a learning algorithm for neural network with elastic output structure," in *Proceedings of the International Symposium on Intelligent Information Systems and Applications*, pp. 271–275, 2009.

[30] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.

[31] Y. Sun, Y. Yuan, and G. Wang, "An OS-ELM based distributed ensemble classification framework in P2P networks," *Neurocomputing*, vol. 74, no. 16, pp. 2438–2443, 2011.

[32] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.

[33] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[34] Z.-L. Sun, K.-F. Au, and T.-M. Choi, "A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 5, pp. 1321–1331, 2007.

[35] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.

[36] Z. Deng, K.-S. Choi, L. Cao, and S. Wang, "T2fela: type-2 fuzzy extreme learning algorithm for fast training of interval type-2 TSK fuzzy logic system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 664–676, 2014.

[37] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, 2001.

[38] S. O. Olatunji, A. Selamat, and A. Abdulraheem, "A hybrid model through the fusion of type-2 fuzzy logic systems and extreme learning machines for modelling permeability prediction," *Information Fusion*, vol. 16, no. 1, pp. 29–45, 2014.

[39] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Microsoft Research Technical Report MSR-TR-98-14, 1998.

[40] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, "Incremental support vector learning: analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, pp. 1909–1936, 2006.

[41] D. M. J. Tax and K.-R. Müller, "A consistency-based model selection for one-class classification," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, pp. 363–366, IEEE Computer Society, Los Alamitos, Calif, USA, August 2004.

[42] A. G. Arkedev and E. M. Braverman, *Computers and Pattern Recognition*, Thompson, Washington, DC, USA, 1966.

[43] L. Manevitz and M. Yousef, "One-class document classification via Neural Networks," *Neurocomputing*, vol. 70, no. 7–9, pp. 1466–1481, 2007.

[44] M. F. Jiang, S. S. Tseng, and C. M. Su, "Two-phasee clustering process for outliers detection," *Pattern Recognition Letters*, vol. 22, no. 6-7, pp. 691–700, 2001.

[45] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985.

[46] D. M. J. Tax and R. P. W. Duin, "Data descriptions in subspaces," in *Proceedings of the International Conference on Pattern Recognition*, vol. 2, pp. 672–675, 2000.

[47] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, vol. 8, no. 3-4, pp. 237–253, 2000.

[48] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Walton Street, Oxford, UK, 1995.

[49] C.-C. Chang and C.-J. Lin, "LIBSVM: a Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.

[50] D. M. J. Tax, *DDtools, the Data Description Toolbox for Matlab*, 2014, http://prlab.tudelft.nl/david-tax/dd_tools.html.

[51] K. Bache and M. Lichman, *UCI Machine Learning Repository*, School of Information and Computer Sciences, University of California, Irvine, Calif, USA, 2013, http://archive.ics.uci.edu/ml.html.

[52] C. J. van Rijsbergen, *Information Retrieval*, Butterworths, London, UK, 1979.