Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2308237, 17 pages
https://doi.org/10.1155/2018/2308237

Research Article

Weighted Domain Transfer Extreme Learning Machine and Its Online Version for Gas Sensor Drift Compensation in E-Nose Systems

Zhiyuan Ma,[1] Guangchun Luo,[1] Ke Qin,[1] Nan Wang,[2] and Weina Niu[1]

[1] *University of Electronics and Technology of China, No. 2006, Xiyuan Avenue, West Hi-Tech Zone, Chengdu, Sichuan 611731, China*
[2] *East China University of Science and Technology, No 130, Meilong Road, Shanghai 200237, China*

Correspondence should be addressed to Guangchun Luo; gcluo.uestc@gmail.com

Received 6 November 2017; Accepted 11 January 2018; Published 12 February 2018

Academic Editor: Dongkyun Kim

Machine learning approaches have been widely used to tackle the problem of sensor array drift in E-Nose systems. However, labeled data are rare in practice, which makes supervised learning methods hard to be applied. Meanwhile, current solutions require updating the analytical model in an offline manner, which hampers their uses for online scenarios. In this paper, we extended Target Domain Adaptation Extreme Learning Machine (DAELM_T) to achieve high accuracy with less labeled samples by proposing a Weighted Domain Transfer Extreme Learning Machine, which uses clustering information as prior knowledge to help select proper labeled samples and calculate sensitive matrix for weighted learning. Furthermore, we converted DAELM_T and the proposed method into their online learning versions under which scenario the labeled data are selected beforehand. Experimental results show that, for batch learning version, the proposed method uses around 20% less labeled samples while achieving approximately equivalent or better accuracy. As for the online versions, the methods maintain almost the same accuracies as their offline counterparts do, but the time cost remains around a constant value while that of offline versions grows with the number of samples.

1. Introduction

The wide spread of wireless sensor environment has provided abundant sources to help improve the convenience and prosperity of human life. Accordingly, the researches have shifted from original construction and routing problems [1–3] to more specific sensory data processing and analyzing tasks [4–6]. With the fast development of sensor technology, E-Nose systems comprised of gas sensor arrays have been widely used in air quality monitors, security check points, and other gas compound identification scenarios. Such devices rely on the direct or indirect reactions between their materials and gas compounds. Taking metal-oxide sensor array as an example, the one that uses chemiresistors will have different electroconductivities when exposed to diverse gases [7]. Partly due to the mechanism of gas sensors, the detection of such reaction may degrade after some time or be exposed in the compound for too long. The phenomenon is called sensor drift, which could hinder the performance of not only sensors themselves, but also the pattern recognition techniques used to determine the compounds. Therefore, an effective way of dealing with the problem is essential for industries.

The cause of drift owes to two sources: the first-order drift, which is due to the interaction process such as long exposure in gas, sensor aging, or poisoning, and the second-order drift, which arises in the experimental setting or system noises [8–11]. Currently, researchers have been trying to solve the problem in several different aspects, and various drift compensation or calibration techniques have been proposed to relieve the problem. Proper choosing of gas sensors that have slow degradation was the first choice of building E-Nose system [12, 13]. With the improvements of materials technologies, durable sensor materials and the proper selection methods have then occurred [7, 14–17]. However, these two types of approaches rely on the resilience of materials or sensor individual differences, which do not tackle the degradation directly and have their limits. Another alternative is to improve postprocessing techniques so that the model continues to work after the degradation happens.

In the postprocessing of the sensor readings, techniques that can track the patterns are suitable for classifying gases when drift occurs. In the process, sensor readings are first preprocessed into multifeatures reflecting different aspects of the readings. Taking the data used in the paper, for example, discrete readings $X[k]$ from a single sensor along time are transformed into a 6-dimensional feature vector X reflecting the steady-state, absorption, and desorption responses. The pattern refers to the distribution of specific gas labeled in the feature space, that is, $P(X \mid Y)$, where Y is the label and X is a n-dimensional feature. The analytical model, which takes X as input, will output the probability of X belonging to a certain label Y. Ideally, the probability of X belonging to its correct label is 1 and 0 for other labels. In the circumstance of drift, the distributions before and after the drift are different; that is, $P(X_{\text{before drift}} \mid Y) \neq P(X_{\text{after drift}} \mid Y)$. In this case, the analytical model that works perfectly before drift is no longer reliable. With current research work in sensor drift compensation, it is commonly accepted that the drift is a slow process and can be compensated by tracking the changes of the sensor readings. Therefore, a detection mechanism based on the features in the postprocessing that can adapt to the distribution changes is required. Ideally, we wish to obtain a mechanism, in which the distributions of specific gas before and after drift are the same; that is, $P(X_{\text{before drift}} \mid Y) = P(X_{\text{after drift}} \mid Y)$.

From the perspective of model learning, sensor drift problem can be viewed as concept drift in which the distribution of gas labels in the feature space changes over time. In the past few decades, some researchers chose to gather different classification models to build a robust one that could resist the drift to some extent [10, 19, 20], while others attempt to map the unknown response to a proper tuned model [21, 22]. Although the two types of models can somehow alleviate the effect of sensor drift, ensemble based method may require proper choosing of the subclassifier, and techniques like transfer learning-based methods require manual labeling of some samples in unknown domain. Additionally, few of the methods considered the imbalanced nature of the samples when building classifiers. In applications, supervised learning is mature and accurate in general, while manual labeling process is time-consuming. Unsupervised approaches require no such labors but are less accurate than their supervised counterparts. To balance between accuracy and time efficiency, more accurate and advanced models with less human involvement are imperative and promising.

In this paper, we are dedicated to building adaptive semisupervised models that allow themselves to train and learn patterns when dataset changes. In particular, we aimed at two different scenarios, of which the first is an offline semisupervised learning with a few manually labeled samples and the second is an online semisupervised learning after labeling samples are selected and unchanged. Our goal is to provide classification models that use less human efforts while maintaining the accuracy of gases identification at a relatively high level. The contributions are threefold.

(i) The samples selected for labeling play key roles in the models. To help select more representative samples while increasing no extra human labor, we proposed a Clustering-Aided Sample Selection (CSS) algorithm.

(ii) The unlabeled samples in DAELM contribute differently to the model training because of the distribution imbalances. Therefore, we proposed Weighted Domain Transfer Extreme Learning Machine (WDTELM), which uses clustering information and a base classifier's output to quantify the effects of unlabeled samples for weighted learning.

(iii) The aforementioned online processing scenario requires the model to update with the arrival of new samples. To save time of retraining the model from scratch, we derived online versions of Target Domain Adaptation Extreme Learning Machine (DAELM_T) and WDTELM, namely, ODAELM and OWDTELM, respectively, which allow the classification to begin with few or none unlabeled samples and update the model in an online manner.

To evaluate the effectiveness of the proposed methods, we conducted different experiments on two aforementioned scenarios. The improvements on the accuracy with less labeled samples verified the feasibility of CSS and the weighted learning mechanism. In addition, ODAELM and OWDTELM have also been proved to be able to update themselves in a time-efficient way and achieve approximately equivalent overall classification performances, when compared with their batch learning versions. For distinguishing purpose, we use DAELM to refer to DAELM_T in L. Zhang and D. Zhang's work [21] in the following parts of the paper.

The remaining of the paper is organized as follows. Section 2 provides some preliminaries on drift compensation and extreme learning machine. Section 3 illustrates the dataset used in the paper and details the CSS, WDTELM, and online learning process. Experimental comparisons on the classification performances have been provided as Section 4. Discussions on the proposed methods and the conclusions are drawn in Sections 5 and 6, respectively. For easy understanding of the terms in the paper, a list containing frequently used abbreviations and corresponding full names is given at the end of the paper.

2. Preliminaries

2.1. Analytical Model for Drift Compensation. Sensor drift is one of the major obstacles that prevent E-Nose systems from being effective for long period of their lifetimes. Therefore, various analytical models have been invented to address the issue.

Ensemble method is one of the most popular methodologies in the field. It uses a group of different classifiers to classify diverse gas compounds so that the sensor drift problem can be mitigated [10, 19, 20]. By doing this, the lifetime of sensor array can be prolonged as well. However, the learning procedures are supervised and require labeling the sample first. Moreover, the method and its variants have assumptions on the gas data, for example, the drift direction remains the same for different gases, which are sometimes not true.

Unsupervised methods such as Sequential Minimal Optimization- (SMO-) based ones have been proved to be effective [23, 24]. Nevertheless, these methods can sometimes mistakenly update the pattern by following the wrong reference class. Component Correction- (CC-) based methods are said to have good results [25–27]. However, they assume that the gases behave in a similar way in the drift process, while the truth is quite the opposite. Drift is a slow process; therefore adaptive methods can be used for it [28]. In [21], the authors proposed a domain adaptive ELM using limited manually labeled samples and semisupervised training procedure to achieve one of the highest accuracies. However, the performance drops rapidly when the number of labeled samples becomes small, and the imbalanced nature of data is not considered either.

It is worth mentioning that some of the techniques, although targeting different problems, can also help transfer the model from one dataset to another using transfer learning [22, 29]. Moreover, instead of detecting and learning the drift problem directly, some of the researchers have contributed by detecting sensors with degrading performance so as to replace them [10, 30, 31]. Although these methods are not included in the discussion of the paper, they do relief the drift by maintaining the performance of E-Nose system at a certain level.

2.2. Extreme Learning Machine. In general, ELM is a three layer feed forward neural network with fully connected nodes between layers. Unlike other neural networks, ELM randomizes the connections between input and hidden layers, while leaving the ones between hidden and output layers to be tuned. The randomness feature of ELM lightens the burden of computing the optimal parameters. Together with the generalized inverse used in the learning process, ELM has been favored as a rapid learning algorithm with good generalization ability [32, 33].

Typical ELM with m hidden layer nodes can be formulated as (1) where H is the corresponding hidden layer output of n training samples (see (2)), β is the output weight matrix, and T is the target.

$$H\beta = T, \quad (1)$$

$$H = \begin{bmatrix} g(a_1, b_1, x_1) & g(a_2, b_2, x_1) & \cdots & g(a_m, b_m, x_1) \\ g(a_1, b_1, x_2) & g(a_2, b_2, x_2) & \cdots & g(a_m, b_m, x_2) \\ \vdots & \vdots & & \vdots \\ g(a_1, b_1, x_n) & g(a_2, b_2, x_n) & \cdots & g(a_m, b_m, x_n) \end{bmatrix}. \quad (2)$$

The fast training speed resides in the fact that only β needs to be determined. To calculate it, ELM tries to solve an optimization problem (3). The solution can be given as $\beta = H^{\dagger}T = (H^TH)^{-1}H^TT$, where $H^{\dagger}$ is the generalized inverse of H, also known as Moore-Penrose generalized inverse. The good generalization of the method is largely attributed to the Moore-Penrose generalized inverse which is used to replace recursive calculation of β in traditional neural network algorithms.

$$\min \quad \|H\beta - T\|^2. \quad (3)$$

Additionally, to better help the algorithm leverage the effect of empirical errors and smallest norm of weights, the optimization problem is modified as (4), where C is a preset parameter or penalty factor.

$$\min \quad \frac{1}{2}\|\beta\|^2 + \frac{C}{2}\|H\beta - T\|^2. \quad (4)$$

For the past decade, there have been a number of variants of ELM. Incremental ELM (IELM) was proposed to change the network structure by adding more hidden layer neurons [34, 35]. Online sequential ELM (OSELM) enables the network to change its output weight matrix so as to adapt to the changes in the data [36]. Kernel trick is not new in machine learning, and the concept was widely used in Supported Vector Machine (SVM). In ELM, the kernel is defined as $K_{\text{elm}} = HH^T$, and, according to Huang et al.'s work in [37], it improves the generalization performance. In recent years, it is also used to speed up the training process [38].

In the past few years, ELM has been regarded as an effective solution for various applications, like active recognition [39], speech emotion recognition [40], and medical data classification [41], to name a few. With the fast developments in big data and distributed systems, there are also literatures dedicated to making the algorithm adaptive to large scale datasets [42] or Map-Reduce framework [43, 44].

3. Semisupervised Methods for Gas Sensor Drift Compensation

In this section, we first performed a brief analysis on the dataset used in the paper, and then proposed CSS to improve the selection of samples for labeling. Subsequently, we aimed at two application scenarios, namely, offline training and online training, and proposed online versions for DAELM and WDTELM, respectively.

3.1. Specification of Dataset. The chemical gas sensor dataset used in the paper has been published on UCI repository [45]. To properly characterize the features of such data, techniques that transform the time-continuous raw data into discrete values are commonly used [46–48]. In this paper, the data are measurements on the conductivity of metal gas sensors array's responses to some gas compounds for continuous 36 months and have been preprocessed using Exponential Moving Average (EMA). Each sample consists of readings from a sensor array of 16 metal gas sensors. For one gas sensor, each sample has 2 steady-state features and 6 dynamic features. In total, there is a 128-dimension feature space for the dataset.

Table 1 is a detailed data distribution of six gas compounds from 10 batches in the dataset. As shown in the table, some of the batches (e.g., batch (1)) have all the six gas compounds while some (e.g., batch (3)) have only five. In addition, the

Table 1: Data distribution [18].

Batch number	Months	Number of samples					
		Ethanol	Ethylene	Ammonia	Acetaldehyde	Acetone	Toluene
(1)	1, 2	83	30	70	98	90	74
(2)	3, 4, 8, 9, 10	100	109	532	334	164	5
(3)	11, 12, 13	216	240	275	490	365	0
(4)	14, 15	12	30	12	43	64	0
(5)	16	20	46	63	40	28	0
(6)	17, 18, 19, 20	110	29	606	574	514	467
(7)	21	360	744	630	662	649	568
(8)	22, 23	40	33	143	30	30	18
(9)	24, 30	100	75	78	55	61	101
(10)	36	600	600	600	600	600	600

number of some gas samples may be 20 times larger than others (e.g., Toluene and Acetone in batch (2)). Moreover, the distributions of the samples in their feature space are also imbalanced.

Figure 1 shows the distributions of 10 batches after Principal Component Analysis (PCA) [49]. The three axes represent the first three dimensions after performing PCA on the dataset. Different colors represent diverse labels. It can be noted that for some labels, such as the purple, the samples cover a large area for most batches while for others, such as the yellow, the samples only expand in small areas. It can also be seen that the data distributions for the classes are scarcely alike. However, the relative position of each class's distribution stays still. For example, the purple ones always stay on the right while the navy ones keep themselves to the left. This phenomenon confirms that using offline trained model on a single batch is highly unrealistic. Nevertheless, some knowledge acquired on one batch may be applied to other batches and semisupervised learning with limited representative samples may help in capturing the differences.

3.2. Clustering-Aided Sample Selection. Due to the fact that sensor data are of various sources and usually redundant, a small size of samples can effectively approximate the distribution of the data with little information loss [50, 51]. These data are called representative data. Therefore, in order to perform semisupervised learning, a group of representative samples should be selected for manual labeling. The selection of the to-be-labeled samples in L. Zhang and D. Zhang's paper [21] uses Kennard and Stone (KS) algorithm, also called SSA in that paper, which is based on Euclidean distance of the features. The effectiveness of L. Zhang and D. Zhang's work confirms that distance-based measurements can be used to distinguish the samples. However, the method treats the samples equally during selection, and it might explain why the method has large degradation when the number of selected samples is small. To further improve the performance and include no extra human labor, we are inspired to use another unsupervised distance-based method before KS to provide additional information for pruning the selection process.

To achieve the goal, clustering is the first choice. By using proper clustering method, we can classify the unlabeled samples in an unsupervised way without extra human involvement. If the clustering is accurate enough, the class information will be provided. Although the exact labels are still unknown, the difference of samples from diverse classes is certain.

Following the intuition, we first examined different clustering strategies in classifying the gas sensor data. Although the following part belongs to the section of performance evaluation, we place it here for better illustration purpose. The performance was evaluated by the accuracy of classification. Even though the clustering method has no concept of classification accuracy, we can define one here for examination purpose. In this article, we assume the label of the majority samples in the cluster is the label of the cluster. Therefore, we can define the performance of the accuracy on each clustering method using (5). $\text{Number}_{\text{pos}}$ represents the number of correctly labeled samples and Total Number is the number of all the samples.

$$\text{Accuracy} = \frac{\text{Number}_{\text{pos}}}{\text{Total Number}}. \tag{5}$$

For each batch of the dataset, we conducted clustering methods on it and summarized the majority label of the samples. Consequently, the samples with the same label in that cluster are considered correctly labeled. In total, 7 built-in hierarchical clustering methods in MATLAB were used in this evaluation process, that is, unweighted average distance (Average), furthest distance (Furthest), centroid distance (Centroid), weighted center of mass distance (Median), shortest distance (Shortest), weighted average distance (Weighted), and inner square distance (Inner). The average accuracies for each methods are as follows: Average: 53.3%; Furthest: 40.8%; Centroid: 52.7%; Median: 51.4%; Shortest: 40.8%; Weighted: 57.7%; and Inner: 69.1%. Although some methods surpass Inner for specific batches, for example, Centroid surpasses Inner in batch (2), the excess part is not significant (around 0.5%), and the method has terrible performance in other batches. To sum up, Inner is the ideal choice among the methods in general. We also conducted the same clustering process using the first 3

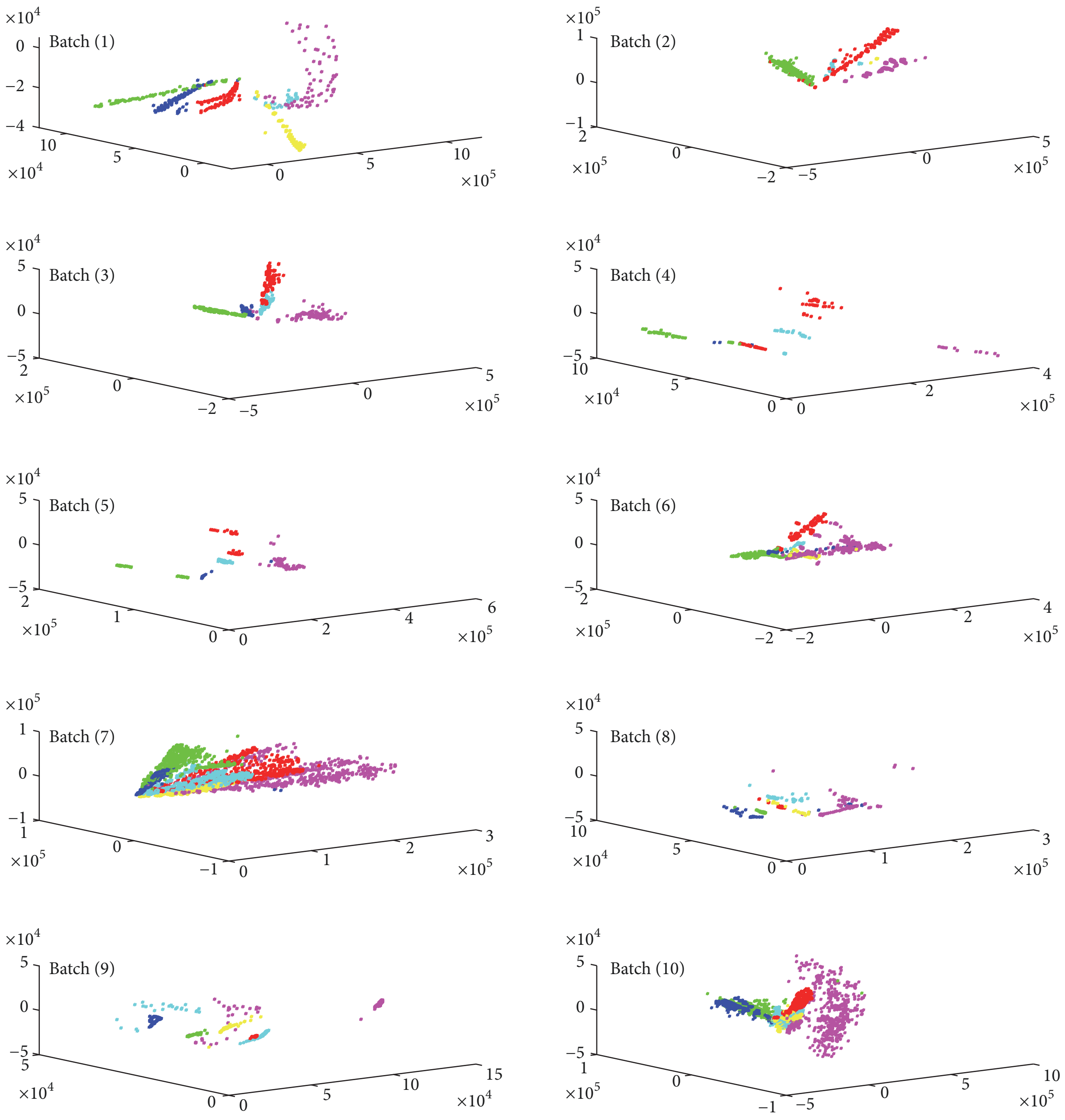


FIGURE 1: Data distributions for 10 batches without normalization. The three axes represent the first 3 features after PCA transformation. Labels of the samples are painted in different colors for better observation.

features after PCA on the dataset and the result remains the same.

Although the clustering performance is relatively good, it is not applicable in real scenario since we cannot decide the exact label for each batch without manually labeling all the samples or at least large amount of samples. Meanwhile, since the data are unlabeled, we cannot find the majority class to help determine the label of the cluster either. Fortunately, we are to solve the problem in a semisupervised way. If we can select the most representative samples from each cluster, the problem can be solved to some extent.

In this paper, CSS uses KS on each cluster to select samples to be labeled. The process can be described as in Figure 2. The cloud-like items represent different clusters and the circles in the figure are the selected samples. The number in the circle represents the sequence of the selection. The dataset in the figure is divided into four clusters and the number of selected samples is 5. Different locations in a cluster represent various values in feature space. In Figure 2, the left upper and right lower clusters have 3 and 4 samples, respectively. As for the other two clusters, the samples could be hundreds, even thousands. As shown in the figure, KS

```
Input:
  $k$ ≔ the number of samples selected for each cluster;
  $C$ ≔ the number of clusters;
Output:
  Labeled ≔ the selected samples;
  Unlabled ≔ the unselected samples;
(1) Clustering $C$ clusters using inner square distance clustering;
(2) for each cluster do
(3)   if the number of samples is less than $k$ then
(4)     Put all the samples into Labeled;
(5)   else
(6)     Calculate the distance between samples in this cluster;
(7)     Selected the two samples with the largest distance and put them in Labeled;
(8)     Initialize $Flag ≔ 2$;
(9)     while $Flag < k$ do
(10)       Find the nearest distances of the remaining samples to the selected ones;
(11)       Choose the one with largest distance and put it in Labeled;
(12)       $Flag = Flag + 1$;
(13)     end while
(14)     Put the unselected samples in Unlabeled;
(15)   end if
(16) end for
(17) return Labeled, Unlabeled;
```

Algorithm 1: Clustering-aided sampling.

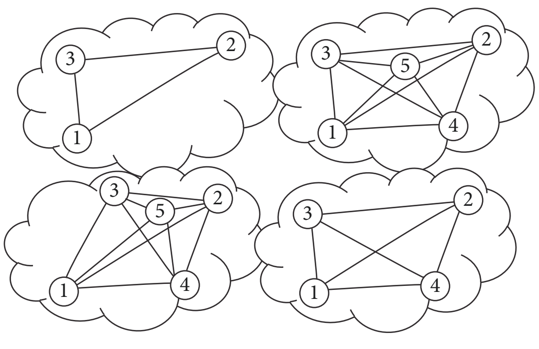


Figure 2: Sample selection in each cluster.

selects the pair with the farthest distance in each cluster, that is, circles labeled 1 and 2, as labeled samples first. Then the sample whose nearest distance to the selected samples is the largest will be selected, that is, circles labeled 3, 4, and 5 in sequence. The process keeps on until the maximum number of samples (k) has been selected in each cluster.

Note that it is possible that the number of samples in some clusters may be less than 5 (see the left upper and right lower clusters in Figure 2). In this case, we directly choose all the samples as selected samples. The pseudocode of CSS is listed in Algorithm 1. For semisupervised learning scenario described in this paper, the exact or estimated number of labels should be set first. As for the dataset in this paper, we set it to 6 for there are 6 different compounds. The maximum value k should be set to be larger than 2 in order for KS to be effective. Although some batches have less than 6 gas compounds, it does not hinder the effectiveness of the method for the selected samples will be labeled. Moreover, the weighted process described later ensures that the unlabeled samples will not be affected by nonexisting label in the batch.

3.3. Weighted Domain Transfer Extreme Learning Machine.

The objective is to train a new classifier on the labeled samples and leverage the effects of the unlabeled ones. In L. Zhang and D. Zhang's paper, the unlabeled samples are treated equally in the function. However, leveraging the effect of unlabeled sample requires distinguishing the difference of correctly and incorrectly classified samples. To be more specific, to help learn a more accurate model, the samples that are being incorrectly classified should weigh less compared with the weight of correct ones. If all the samples were treated equally, the negative effects of the wrongly classified samples would be amplified which would cause the learning to follow a wrong pattern and reduce the classification accuracy. This explains why the accuracy of DAELM degrades quickly when the labeled samples are few. In this paper, our proposed method intends to employ weighted learning to emphasize the effects of the samples that are less likely to be incorrectly classified by the base classifier. Therefore, the optimization problem becomes (6) by incorporating a sensitive matrix w.

$$\min \quad \frac{1}{2}\left\|\beta_T\right\|^2 + \frac{C_t}{2}\left\|H_T\beta_T - T_T\right\|^2 + \frac{C_{Tu}}{2}\left\|w \circ \left(H_{Tu}\beta_T - H_{Tu}\beta_S\right)\right\|^2 . \tag{6}$$

In (6), H_T and H_{Tu} are the hidden layer outputs of the labeled and unlabeled samples, respectively. T_T is the labels of the labeled samples. C_T and C_{Tu} are two preset parameters for regularization purpose. β_S is the output weight

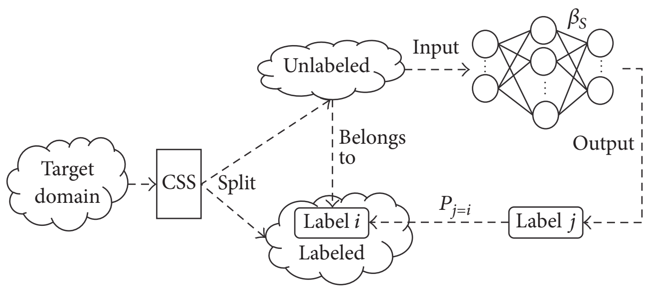


Figure 3: Demonstration of determining the effects of different unlabeled samples.

of ELM trained in source domain (base ELM) and β_T is the one for target domain classifier to be learned. Note that the labeled samples are fixed and reliable with their labels manually examined and determined. For unlabeled samples, the third term wishes to learn their information based on the output from a classifier trained from source domain. However, β_S is not 100% reliable and the labels could be wrong. Therefore, the learning algorithm should learn the information of the samples whose labels are most unlikely to be wrong and ignore the ones that are most likely to have incorrectly classified labels. In this paper, we solve the problem in the optimization by adding w in the third term in which the symbol $\circ$ represents Hadamard product.

$$w_i = \begin{bmatrix} P_1 & P_2 & 0 & P_4 & 0 & 0 \end{bmatrix}. \tag{7}$$

In detail, w has the same number of rows as H_{Tu} and each row represents the probabilities of the sample belonging to specific labels. For example, let there be 3 labels, namely, 1, 2, and 4, in the cluster which the ith sample belongs to. The number of labels in total is 6. Equation (7) is an example of the ith row in w. The values in columns 1, 2, and 4 are the probabilities of ith sample belonging to each class, denoted by P_1, P_2, and P_4. Since there are no more labels in the cluster, the rest of the columns is 0.

The ideal value of w is unknown for semisupervised scenario for we do not know the exact labels of the unlabeled samples. However, we can estimate it with the information we collected from clustering process. In the labeling process, the selected samples are labeled with certain label(s). It is deterministic. Considering the fact that, in clustering, the more closer the distance is, the more likely the samples share the same label. We can further extend the idea to the following: the more closer the samples are to certain labeled sample, the more likely they belong to its labels. The demonstration of the idea can be viewed in Figure 3. With the help of CSS, the target domain is split into labeled and unlabeled sets. The labeled samples are representative and we assume that the labels of the unlabeled belong to the ones of the labeled. Meanwhile, we partially trust β_S trained from the source domain. Therefore, the output using β_S, say label j, may or may not be true. Note that the representative samples are chosen based on the distances among samples. Considering the fact that the real label is one of the labeled samples, say label i, under the aforementioned assumption, the probability of $i = j$ can be calculated based on the distance.

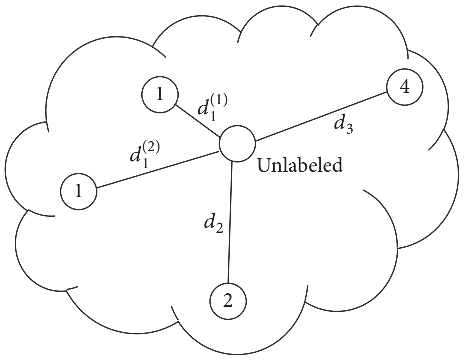
In this paper, we use the reciprocal of the distance to a certain labeled sample as the degree to its label. If more than one selected sample in a cluster belongs to the same label, we use the one with smallest distance. The probability of an unlabeled sample belonging to a given label can be calculated based on all the degrees to all the labels in the cluster.

Figure 4 is the example of calculating each value in (7). There are 3 different labels in the example. The center circle is the unlabeled sample that requires to be estimated and circles labeled 1, 2, and 4 are the selected samples with their labels being the numbers. The lines are the distances between the unlabeled and the selected samples, tagged by d_i, where i is the label number. In this process, for a specific unlabeled samples in one cluster, we assume that it can only belong to the labels of the selected samples. If more than one selected sample is of the same label, that is, 1 in the figure, we choose the smaller or smallest one; that is, d_1 is the smaller one between $d_1^{(1)}$ and $d_1^{(2)}$. By doing so, we can calculate the distance of the unlabeled sample to each label, written as $\text{dist}(\text{unlabeled}, i), i = 1, 2, 4$. Subsequently, the probability of the unlabeled belonging to each label j is P_j where $j \in \{1, 2, 4\}$. For those labels (3, 5, and 6) that do not appear in this cluster, we set the probability P_k to 0 where $k \in \{3, 5, 6\}$. Eventually, we have each estimated value of (7). The calculation of probabilities is summarized as (8). In the same way, we can calculate each row of w.

$$P_j = \begin{cases} \dfrac{(1/d_j)}{\left(\sum_n^{i=1} 1/d_i\right)}; & \text{if } j \in \text{the labels of selected samples} \\ 0; & \text{otherwise.} \end{cases} \tag{8}$$

We call w sensitive matrix in this paper, and the pseudocode of calculating w is given in Algorithm 2. For each cluster generated in the clustering phase, the distances between unlabeled and labeled samples are calculated within the cluster. w is formed with exact number of labels in the dataset and each value in a column represent the probability of an unlabeled sample belonging to a specific label. The calculation ensures that, for each cluster, the unlabeled samples only belong to the labels of the labeled ones in the cluster, which makes the probability of belonging to the labels that are not in the cluster 0, and the sum of each row equals 1. In CSS, the number of clusters for each batch is set to 6, which could make the samples belonging to a specific label be split into two or more clusters. In this case, the sensitive matrix calculation can still determine the probability that each of the unlabeled samples belonging to a certain label for the split cluster will have its representative samples of the same label.

With the sensitive matrix calculated, we can further train the classifier in a weighted learning way. To solve (6), we first calculate the gradient with respect to β_T as (9). Let $w \circ H_{Tu}$ be



dist(unlabeled, 1) $= d_1 = \min(d_1^{(1)}, d_1^{(2)})$;
dist(unlabeled, 2) $= d_2$;
dist(unlabeled, 4) $= d_4$;

$$P_j = \frac{1/d_j}{\sum_{i=1,2,4}(1/d_i)}, \quad j = 1, 2, 4,$$

$$P_k = 0, \quad k = 3, 5, 6.$$

Figure 4: The probability of a sample belonging to a certain label.

```
Input:
  Labeled := the selected samples in a cluster;
  Unlabeled := the unlabeled samples in a cluster;
Output:
  W := the sensitive matrix samples;
(1) Initialize W to a zero matrix;
(2) for each samples $S \in$ Labeled do
(3)   Find unique labels as Labels
(4)   for each value $L \in$ Labels do
(5)     Find the samples with label $L \in$ Labeled as TempSet;
(6)     Calculate the distances between S and the samples in TempSet;
(7)     Set $d_j$ to the nearest distance;
(8)     Store $1/d_j$ in W;
(9)   end for
(10)  Replace the value with probability in W using (8);
(11) end for
(12) return W;
```

Algorithm 2: Sensitive matrix calculation.

H_u. We can solve the equation by setting (9) to zero and we get (10), in which I is the identity matrix.

$$\begin{aligned}\frac{\delta L}{\delta \beta_T} &= \beta_T + C_T\left(H_T^T H_T \beta_T - H_T^T T_T\right) \\ &\quad + C_{Tu}\left(w \circ H_{Tu}\right)^T \left(w \circ H_{Tu}\right)\beta_T \\ &\quad - C_{Tu}\left(w \circ H_{Tu}\right)^T \left(w \circ H_{Tu}\right)\beta_S,\end{aligned} \tag{9}$$

$$\begin{aligned}&\left(I + C_T H_T^T H_T + C_{Tu} H_u^T H_u\right)\beta_T \\ &\quad = C_T H_T^T T_T + C_{Tu} H_u^T H_u \beta_S.\end{aligned} \tag{10}$$

If H_T has more rows than columns, that makes (6) an overdetermined problem and the least square solution is unique. According to the work in [38, 52], a special RBF function, that is, Gaussian RBF function, ensures that the inverses of $H^T H$ and HH^T exist. Therefore, we use this type of function in the paper. Note that C_T and C_{Tu} are positive. By using the formulas of generalized inverse of sum of matrices [53], it can be verified that $I + C_T H_T^T H_T + C_{Tu} H_u^T H_u$ has a unique generalized inverse which is also the inverse. In this case, β_T can be written as follows:

$$\begin{aligned}\beta_T &= \left(I + C_T H_T^T H_T + C_{Tu} H_u^T H_u\right)^{-1} \\ &\quad \cdot \left(C_T H_T^T T_T + C_{Tu} H_u^T H_u \beta_S\right).\end{aligned} \tag{11}$$

For the case where H_T has less rows than columns, (6) becomes an underdetermined problem. In L. Zhang and D. Zhang's work [21], they solved the problem by using Lagrange multiplier. The method is equal to adding an assumption that β_T is the linear combination of the columns of H_T and H_{Tu}; that is, $\beta_T = H_T \alpha_T + H_{Tu} \alpha_{Tu}$. It is hard to determine if it applies to all cases. However, since the two separate cases are divided based on the rows and columns of H_T, the unlabeled term H_{Tu} should be excluded. Therefore, in this paper, we assume that β_T is a linear combination of the columns of H_T, written as (12), so as to get the unique solution. Then the problem becomes solving α.

$$\beta_T = H_T^T \alpha. \tag{12}$$

```
Input:
    $TrSet$ := the training data set;
    $TSet$ := the unlabeled data set;
    $C$ := the number of clusters;
    $N$ := the number of hidden layer nodes;
Output:
    $\beta_T$ := the output weight matrix for target domain classifier;
(1) Initialize a base classifier ELM with $N$ nodes using $TrSet$;
(2) Set the output weight to $\beta_S$;
(3) Clustering $TSet$ into $C$ clusters;
(4) Select the label and unlabeled samples as Labeled and Unlabeled using Algorithm 1;
(5) for each cluster do
(6)     Update the weight matrix $W$ using Algorithm 2.
(7) end for
(8) Initialize a new ELM with $N$ nodes;
(9) if the number of samples in Labeled > $N$ then
(10)    Calculate the output weight matrix $\beta_T$ using (11);
(11) else
(12)    Calculate the output weight matrix $\beta_T$ using (16);
(13) end if
(14) return $\beta_T$;
```

ALGORITHM 3: Weighted Domain Transfer Extreme Learning Machine.

By multiplying $(H_T^T H_T)^{-1} H_T$ on both sides of (10), we can get

$$\begin{aligned}&\left(H_T^T H_T\right)^{-1} H_T\beta_T + C_T\left(H_T^T H_T\right)^{-1} H_T H_T^T H_T\beta_T \\ &\quad + C_t u\left(H_T^T H_T\right)^{-1} H_T H_u^T H_u^T \beta_T \\ &= C_T\left(H_T^T H_T\right)^{-1} H_T H_T^T T_T \\ &\quad + C_{Tu}\left(H_T^T H_T\right)^{-1} H_T H_u^T H_u \beta_S.\end{aligned} \tag{13}$$

In order to solve α, we can substitute (12) into (13) and get

$$\begin{aligned}&\left(I + C_T H_T^T H_T + C_{Tu}\left(H_T^T H_T\right)^{-1} H_T H_u^T H_u H_T^T\right)\alpha \\ &= C_T T_T + C_{Tu}\left(H_T^T H_T\right)^{-1} H_T H_u^T H_u \beta_S.\end{aligned} \tag{14}$$

For simple illustration purpose, let P be $H_T^T H_T$ and Q be $H_T H_u^T$. Similarly, we can verify that $I + C_T P + C_{Tu} P^{-1} Q Q^T$ has inverse. Subsequently, we can get α as (15), and β_T can be written accordingly as (16).

$$\begin{aligned}\alpha &= \left(I + C_T P + C_{Tu} P^{-1} Q Q^T\right)^{-1} \\ &\quad \cdot \left(C_T T_T + C_{Tu} P^{-1} Q H_u \beta_S\right),\end{aligned} \tag{15}$$

$$\begin{aligned}\beta_T &= H_T^T\left(I + C_T P + C_{Tu} P^{-1} Q Q^T\right)^{-1} \\ &\quad \cdot \left(C_T T_T + C_{Tu} P^{-1} Q H_u \beta_S\right).\end{aligned} \tag{16}$$

The pseudocode for WDTELM with CSS is listed as Algorithm 3. Before the training begins, we set the number of clusters (C) to the number of gases in the dataset. The algorithm trains a new ELM with a preset number of hidden layers using the source domain samples (line (1)). For target domain denoted by $TSet$, it uses CSS to select a group of labeled samples (lines (3)–(7)) and then initializes a new ELM network with the same hidden layer neurons (line (8)). The output weight β_T of the ELM in target domain is then calculated based on the aforementioned cases (lines (9)–(13)).

3.4. Online Domain Transfer Extreme Learning Machine. In order for DAELM and WDTELM to be applicable, an initial set of data is required. However, in real application scenario, the data may not be accessible in full. More common situation is that data come in a one-by-one or chunk-by-chunk manner. To retrain the classifier in an offline manner would be unrealistic and time-consuming whenever the data come. As DAELM and WDTELMT are all based on batch training and updating, the problem remains the same. Therefore, online learning process is needed.

In this section, we consider a simple online scenario where the labeled samples have been determined or provided beforehand. Then the unlabeled samples are fed into the model in an one-by-one or chunk-by-chunk manner and we wish to use the semisupervised method in previous subsections without retraining the network from scratch.

The demonstration of ODTELM is provided in Figure 5. The unlabeled samples are organized in a sequence as the left rectangle. The target classifier is initialized with the labeled sample using (4). For the online learning phase, we wish to update the target classifier using current β_T, the incremental value δh, and some intermediate result(s).

In this case, we have insufficient data to perform clustering before training. However, given the labeled samples, we can still calculate the probability of each unlabeled sample. To derive the proper formulas, we begin with unweighted

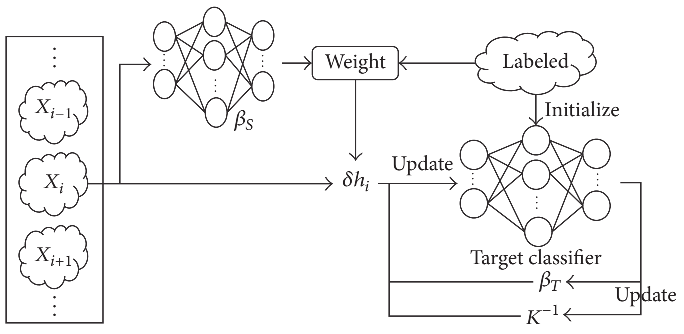


FIGURE 5: Demonstration of online domain transfer extreme learning machine.

version, that is, DAELM. In this case, we use the same objective function as follows:

$$\min \quad \frac{1}{2}\|\beta_T\|^2+\frac{C_t}{2}\|H_T\beta_T-T_T\|^2 + \frac{C_{Tu}}{2}\|H_{Tu}\beta_T-H_{Tu}\beta_S\|^2. \tag{17}$$

Assume an incremental batch δX is added to the unlabeled data in the target domain and let the corresponding hidden layer output be δh. The output weight $\beta_T^{(k+1)}$ can be calculated in the aforementioned cases as (18), where $P = H_T H_T^T$ and $Q = H_T H_{Tu}^T$.

$$\beta_T = \begin{cases} \left(I - C_T H_T^T H_T + C_{Tu} H_{Tu}^T H_{Tu}\right)^{-1}\left(C_T H_T^T T_T + C_{Tu} H_{Tu}^T H_{Tu}\beta_S\right), & H_T \text{ has more rows} \\ H_T^T\left(I + C_T P + C_{Tu} P^{-1} Q Q^T\right)^{-1}\left(C_T T_T + C_{Tu} P^{-1} Q H_{Tu}\beta_S\right), & H_T \text{ has more columns.}\end{cases} \tag{18}$$

For the case where H_T has more rows than columns, define the intermediate result K as follows:

$$K = I + C_T H_T^T H_T + C_{Tu} H_{Tu}^T H_{Tu}. \tag{19}$$

For simple illustration purpose, let Right $= C_T H_T^T T_T + C_{Tu} H_{Tu}^T H_{Tu}\beta_S$. Let the intermediate result for current ELM be K_k, and the output weight matrix be $\beta_T^k = K_k^{-1}\text{Right}_k$. When new sample δX arrives, K_{k+1} and Right_{k+1} change into

$$\begin{aligned} K_{k+1} &= I + C_T H_T^T H_T + C_{Tu}\left(H_{Tu}^T H_{Tu} + \delta h^T \delta h\right) \\ &= K_k + C_{Tu}\delta h^T \delta h, \\ \text{Right}_{k+1} &= \text{Right}_k + C_{Tu}\delta h^T \delta h \beta_S. \end{aligned} \tag{20}$$

Based on Sherman-Morrison-Woodbury formula, the inverse of K_{k+1} can be obtained as follows:

$$\begin{aligned} K_{k+1}^{-1} &= \left(K_k + C_{Tu}\delta h^T \delta h\right)^{-1} \\ &= K_k^{-1} \\ &\quad - C_{Tu} K_k^{-1}\delta h^T\left(I + C_{Tu}\delta h K_k^{-1}\delta h^T\right)^{-1}\delta h K_k^{-1}. \end{aligned} \tag{21}$$

Note that $\beta_T^{k+1} = K_{k+1}^{-1}\text{Right}_{k+1}$. By multiplying $K_k K_k^{-1}$ before Right_k in (20), we can obtain the formula for β_T^{k+1} as follows:

$$\begin{aligned} \beta_T^{k+1} &= K_{k+1}^{-1}\left(K_k K_k^{-1}\text{Right}_k + C_{Tu}\delta h^T \delta h \beta_S\right) \\ &= K_{k+1}^{-1}\left(K_k \beta_T^k + C_{Tu}\delta h^T \delta h \beta_S\right) \\ &= K_{k+1}^{-1}\left(\left(K_{k+1} - C_{Tu}\delta h^T \delta h\right)\beta_T^k + C_{Tu}\delta h^T \delta h \beta_S\right) \\ &= \beta_T^k - K_{k+1}^{-1} C_{Tu}\delta h^T \delta h\left(\beta_T^k - \beta_S\right). \end{aligned} \tag{22}$$

For the case where H_T has more columns than rows, we can write intermediate result K_{k+1} as (23). Let $Q_{k+1}Q_{k+1}^T$ and Right_{k+1} be (24) and (25), respectively.

$$\begin{aligned} K_{k+1} & \\ &= \left(I + C_T P + C_{Tu} P^{-1} Q_k Q_k^T + C_{-1} P^{-1} H_T \delta h^T \delta h H_T^T\right) \\ &= K_k + C_{Tu} P^{-1} H_T \delta h^T \delta h H_T^T, \end{aligned} \tag{23}$$

$$\begin{aligned} Q_{k+1}Q_{k+1}^T &= H_T\begin{bmatrix} H_{Tu}^T & \delta h^T \end{bmatrix}\begin{bmatrix} H_{Tu} \\ \delta h \end{bmatrix} H_T^T \\ &= Q_k Q_k^T + H_T \delta h^T \delta h H_T^T, \end{aligned} \tag{24}$$

```
Input:
    $\beta_T^k$ ≔ the output weight matrix of ELM for target domain;
    $N$ ≔ the number of hidden layer neurons;
    $K_k^{-1}$ ≔ the intermediate result;
    $Labeled$ ≔ the labeled samples;
    $U$ ≔ the increment of unlabeled samples;
Output:
    $\beta_T^{k+1}$ ≔ the updated output weight matrix;
    $K_{k+1}^{-1}$ ≔ the updated intermediate result;
(1) Calculate the hidden layer output for $Labeled$ as $H_T$;
(2) Calculate the hidden layer output for $U$ as $\delta h$;
(3) if the rows of $H_T > N$ then
(4)     Calculate $K_{k+1}^{-1}$ using (21);
(5)     Calculate $\beta_T^{k+1}$ using (22);
(6) else
(7)     Calculate $K_{k+1}^{-1}$ using (26);
(8)     Calculate $\beta_T^{k+1}$ using (27);
(9) end if
(10) return $\beta_T^{k+1}, K_{k+1}^{-1}$;
```

Algorithm 4: Online Domain Transfer Extreme Learning Machine update.

$$\begin{aligned}\text{Right}_{k+1} &= C_T T_T + C_{Tu} P^{-1} H_T \left(H_{Tu}^T H_{Tu} + \delta h^T \delta h\right) \beta_S \\ &= \text{Right}_k + C_{Tu} P^{-1} H_T \delta h^T \delta h \beta_S.\end{aligned} \tag{25}$$

For illustration purpose, let $\delta k = \delta h H_T^T$. Similarly, K_{k+1}^{-1} can be derived as (26) based on Sherman-Morrison-Woodbury formula.

$$K_{k+1}^{-1} = K_k^{-1} - K_k^{-1} C_{Tu} P^{-1} \left(I + \delta k^T \delta k K_k^{-1} C_{Tu} P^{-1}\right)^{-1} \cdot \delta k^T \delta k K_k^{-1}. \tag{26}$$

Consequently, the output weight β_T^{k+1} can be derived as follows:

$$\begin{aligned}\beta_T^{k+1} &= H_T^T K_{k+1}^{-1} \text{Right}_{k+1} = H_T^T K_{k+1}^{-1} \left(\text{Right}_k \right. \\ &\left. + C_{Tu} P^{-1} H_T \delta h^T \delta h \beta_S\right) = H_T^T K_{k+1}^{-1} \left(K_k K_k^{-1} \text{Right}_k \right. \\ &\left. + C_{Tu} P^{-1} H_T \delta h^T \delta h \beta_S\right) \\ &= H_T^T K_{k+1}^{-1} \left(\left(K_{k+1} - C_{Tu} P^{-1} H_T \delta h^T \delta h H_T^T\right) \right. \\ &\left. \cdot K_k^{-1} \text{Right}_k + C_{Tu} P^{-1} H_T \delta h^T \delta h \beta_S\right) = \beta_T^k \\ &- C_{Tu} H_T^T K_{k+1}^{-1} P^{-1} H_T \delta h^T \delta h \left(\beta_T^k - \beta_S\right).\end{aligned} \tag{27}$$

The pseudocode for the updating procedure of ODAELM is given in Algorithm 4. Whenever a new sample arrives, the algorithm takes the intermediate result K_k and output weight β_k calculated from last update and updates current intermediate result and output weight in two cases. The calculation saves time by using results calculated in previous updates, so the training is time-saving, especially when the unlabeled set becomes too large. It is essential to seamless service especially for scenarios like antiterrorism, security checkpoint, and so on.

Note that the formulas are derived from DAELM. If we are to perform online learning for WDTELM, taking Figure 5 for example, the weight w can be calculated by comparing the output from ELM with β_S and the labeled samples. Let δw be the weight for δX. Then, we can simply switch H_{Tu} and δh in this subsection to $w \circ H_{Tu}$ and $\delta w \circ \delta h$, respectively, and the formulas still stand.

4. Performance Evaluation

4.1. Experimental Setup. The experiments in the paper were conducted in MATLAB on a Linux Workstation with an E5 2.6-GHz CPU and 32-GB RAM. We followed the setup in [21]. The hidden layer neurons of the ELM network were by default 1000, with RBF function being their activation function. The coefficients C_S, C_T, and C_{Tu} were set to be 0.001, 100, and 0.001, respectively. We also tested hidden layer neurons with 100 hidden layers to show the performance of the proposed methods.

For WDTELM and its counterparts, the experiments used the current batch as source domain and the next batch as target domain. For example, batch one was used as source domain at first and batch two was target domain. After classification of batch two was finished, the source domain became batch two and target domain would be batch three. The performance was evaluated by the classification accuracy using (5). The proposed method was compared with SVM with rbf kernel (SVM-rbf), ELM with rbf activation function (ELM-rbf), ELM based ensemble methods (Ensemble ELM), SVM based ensemble method (Ensemble SVM), and DAELM with diverse numbers of selected samples. The ensemble methods trained subclassifiers on each batch using corresponding algorithms and combined with previous learned

Table 2: Comparisons of classification accuracy in 9 batches.

Batch number	Accuracy (%)							
	Ensemble SVM	Ensemble ELM	SVM -rbf	ELM -rbf	DAELM (30)	DAELM (50)	WDTELM (<30)	WDTELM (<48)
(2)	77.4	75.3	58.5	76.8	85.5	96.2	90.2	98.2
(3)	74.8	76.9	66.9	75.4	94.0	98.8	90.1	94.6
(4)	72.0	64.5	74.5	69.8	100	100	100	100
(5)	65.9	96.0	44.2	56.6	99.5	100	98.9	99.5
(6)	71.3	71.3	58.3	62.0	73.4	82.1	95.3	98.5
(7)	55.1	68.8	68.9	68.0	64.3	79.8	78.1	84.7
(8)	42.9	81.8	60.5	87.7	94.6	97.6	94.6	97.3
(9)	56.6	49.6	50.9	40.0	100	100	99.1	100
(10)	32.9	49.4	18.4	19.3	41.5	55.8	62.6	64.6
Ave.	64.1	72.5	55.7	61.7	83.8	90.1	89.9	93.0

Table 3: Comparisons of DAELM and WDTELM using different numbers of selected samples.

Batch number	DAELM (%)						WDTELM (%)					
	$k = 20$	$k = 25$	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k \leq 12$	$k \leq 18$	$k \leq 24$	$k \leq 30$	$k \leq 36$	$k \leq 42$
(2)	66.7	78.1	85.5	85.1	93.2	92.8	82.8	82.5	84.0	90.2	95.7	95.7
(3)	76.2	90.3	94.0	96.3	96.3	96.9	75.8	83.0	90.5	90.1	92.4	92.4
(4)	100	100	100	100	100	100	87.0	92.5	100	100	100	100
(5)	99.4	99.4	99.5	100	100	100	94.4	99.5	99.5	98.9	99.5	98.9
(6)	54.6	59.3	73.4	75.7	77.5	80.9	85.5	88.7	94.1	95.3	96.5	98.2
(7)	62.6	60.4	64.3	68.3	72.7	74.1	57.5	64.9	79.2	78.1	80.6	87.0
(8)	69.4	94.8	95.5	95.9	96.5	96.9	71.4	77.2	88.1	94.6	95.9	97.3
(9)	100	100	100	100	100	100	92.1	93.2	95.5	99.1	98.9	99.8
(10)	37.0	38.8	41.5	47.9	51.6	52.1	55.8	52.0	58.6	62.6	57.7	62.2
Ave.	73.9	80.1	83.8	85.5	87.6	88.2	76.9	81.5	87.7	89.9	90.8	92.4

subclassifiers to form a compact classifier. For example, when batch (3) was target domain, targets 1 and 2 would be used to train their subclassifiers, respectively. The weight for each subclassifier followed the work in [20], which is the training accuracy on the corresponding batch.

For online learning, the source and target domains were divided in the same way as described in the previous paragraph. The difference was that, after representative samples were fixed, the unlabeled samples were tested in sequence and fed to the classification model in a one-by-one manner for updating. The overall accuracy was recorded when all the unlabeled samples were used for updates.

4.2. Performance Evaluation. Table 2 shows the overall classification performances of different models used in the experiments. The last row labeled "Ave." refers to average accuracy for each method which applies to Tables 2, 3, 5, and 6 Note that the performance of WDTELM surpasses all its counterparts except for batch (3) where it is around 3% to 4% less than that of DAELM. For batch (5) and (8), although the accuracy of DAELM beats WDTELM by 0.1% to 0.7%, the difference is not distinctive. As for ELM and SVM, the two methods have over 20% accuracy lower than those of DAELM and WDTELM. Ensemble based methods have better classification accuracy in general. This is due to the increasing training data used for generating subclassifiers. Although the method can somehow alleviate the drift, it does not perform as well as semisupervised methods do in the paper. Although WDTELM has slightly lower performance than that of DAELM for specific batches, say batch (3), in an overall point of view, it beats DAELM. Taking batches (7) and (10), where DAELM has its classification accuracy lower than other batches, for example, WDTELM has 5% to 10% accuracy increase. As for the other batches, WDTELM has slightly better performance. Therefore, in general, WDTELM captures the changes in the data more accurately and therefore can better help the gas sensors bounce back from degradation caused by drift problem. Moreover, the labeled samples for WDTELM are chosen with k equal to 5 and 8, and the selection process will choose no more than 30 and 48 samples, respectively. Therefore, the selected samples for WDTELM are no more than that of DAELM. For real application scenario, the manual labeling process is more time-consuming than that of training ELM from scratch. In this sense, WDTELM is more time-saving than DAELM is.

The overall performance shown in Table 2 confirms that proper choice of selected samples helps the model to achieve better classification performance. To better show the performance improvements of WDTELM over DAELM, we

Table 4: Average classification accuracy.

	Average accuracy (%)						
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
ELM-rbf	44.8	41.5	42.9	46.0	47.1	50.8	51.0
SVM-rbf	69.7	72.0	73.5	74.4	76.5	77.9	79.0
DAELM	62.7	72.9	79.6	84.4	86.0	89.1	88.9
WDTELM	76.1	80.6	86.9	88.2	88.6	89.2	91.5

Table 5: Classification accuracy on ODAELM.

Batch number	Accuracy of ODAELM (%)						
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
(2)	67.5	79.6	83.3	87.5	94.4	93.0	96.4
(3)	78.8	94.6	97.7	97.8	97.7	99.0	97.9
(4)	100	100	100	100	100	100	100
(5)	99.4	99.4	100	100	100	100	100
(6)	54.2	72.0	75.3	79.1	78.6	81.9	84.9
(7)	65.3	65.5	71.0	73.4	80.3	81.4	81.7
(8)	67.6	93.8	95.2	96.2	96.5	96.9	97.2
(9)	100	100	100	100	100	100	100
(10)	40.1	40.2	46.2	48.0	53.2	52.6	51.8
Ave.	74.7	82.8	85.4	86.9	88.9	89.4	90.0

Table 6: Classification accuracy of OWDTELM.

Batch number	Accuracy of OWDTELM (%)						
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
(2)	81.7	82.6	83.2	88.9	95.3	94.8	98.0
(3)	74.9	83.0	91.4	88.8	94.6	92.4	89.3
(4)	87.0	99.4	100	100	100	100	100
(5)	94.4	99.5	99.5	99.0	99.5	99.0	99.5
(6)	83.4	89.0	95.7	95.4	98.6	97.4	98.2
(7)	56.8	64.2	79.3	78.7	80.8	84.7	86.5
(8)	71.8	77.9	82.9	95.9	96.6	97.3	98.3
(9)	92.1	93.4	95.1	98.9	99.6	99.8	100
(10)	45.1	54.5	55.3	61.7	59.1	61.8	65.2
Ave.	76.4	82.6	86.9	98.7	91.6	91.9	92.8

experimented on different numbers of selected samples for DAELM and WDTELM. Table 3 shows the performance of the two methods using 6 different numbers of selected samples, respectively. The k value represents the number of selected samples and the rows show the classification accuracies for different batches, among which the last row shows the average classification performance of each k value. Note that for DAELM k is the exact number of selected samples, while for WDTELM it is the upper bound because of CSS process.

In general, it can be noted that the average performance increases as the number k increases and it is almost the same for all rows except for some disturbances such as batch (2) with $k = 45$. The increasing of selected samples almost guarantees the performance increase. However, the price is the manual labeling process. If the process was quick, there would be no need for classification models. Therefore, the maximum performance with minimum labeling is an ideal solution for semisupervised learning scenario. In the table, the classification accuracy of WDTELM increases around 10% in those settings where DAELM has low performance before $k = 30$, for example, $k = 20$ for DAELM versus $k < 18$ for WDTELM in batch (2). The performance of DAELM has its almost maximum classification performance after 30. As an exception, DAELM surpasses WDTELM in batch (3) with around 4% after $k = 40$. The difference is not large compared with other batches where DAELM has low accuracy, and the accuracies of both methods are over 90% as k increases. Therefore, we consider them both feasible for application scenario. In general, together with Table 2, DAELM does not surpass WDTELM unless the selected samples are large, say 50. If manually determining the gas type takes 1 h for one sample, DAELM would take 10 more hours before its accuracy outperforms WDTELM for some batches.

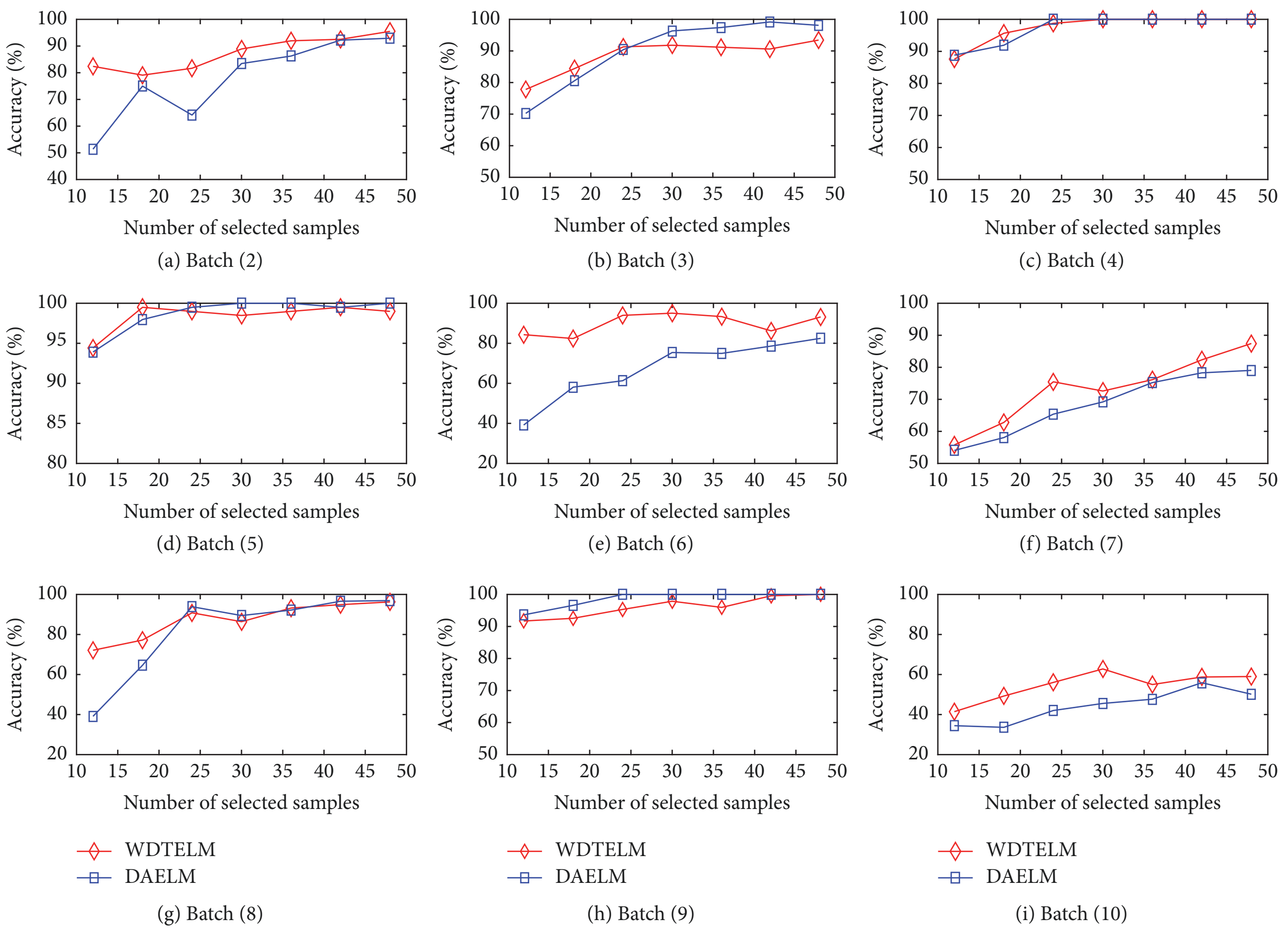


Figure 6: Classification performance of DAELM and WDTELM under 100 hidden layer nodes. The number of labeled samples is chosen from [12, 18, 24, 30, 36, 42, 48] and the overall classification accuracies are recoded accordingly.

To better illustrate the differences of the performance of DAELM and WDTELM, we conducted experiments on the same number (k) of selected samples for each batch. The number of hidden layer is set to 100 so as to observe the effects of a smaller ELM network. The results can be found as in Figure 6, in which the red curves represent the classification performance of WDAELM and the blue ones are of DAELM. As can be observed from the figures, the red curves have significant accuracy increase compared with the blue ones in batches (2), (6), (7), and (10). The maximum amount of increase reaches over 20%; for example, $k = 24$ in batch (6). For batches (4), (5), and (8), the lines intertwine together, which indicate that the performance is similar. For batches (3) and (9), DAELM is better than WDTELM with around 5% to 10% accuracy increase. However, the performance begins to overlap after 42. Taking batch (9), for example, the difference after $k = 42$ is hard to distinguish. The reason for this phenomenon may be that the distribution of the clusters in these two batches changes more seriously than other batches which makes it hard for CSS in WDTELM to capture representative ones in some clusters. Fortunately, the performance difference can be decreased by increasing k. Meanwhile, for the part where DAELM beats WDTELM, the accuracies for both methods are relatively high, say over 90%. Note that, when the numbers of selected samples are small, for example, 12 and 18, the performance of DAELM is terrible. For example, for batches (2), (6), and (8), the accuracies drop below 60%, while WDTELM maintains its accuracy above 70%.

To further compare the significance of the two methods regarding their classification accuracy, we performed Friedman's test. Their performance is grouped by their k values, that is, each k has the corresponding average accuracies of the two methods on 9 batches. Therefore, there are 7 groups in total. The p value of Friedman's test is $9.0214e^{-18}$ which means the classification performances are significantly unlike each other. In order to show which one surpasses the other, we summarized the average classification accuracy as Table 4, together with other batch learning algorithms. To further show that the improvements are not only due to the labeled samples in target domain, we train ELM and SVM in a way that the labeled samples in target domain are combined with source domain for training the classifiers. Although the average accuracy for SVM-rbf is higher than the one in Table 2, it is still over 10% lower than those of DAELM and WDTELM. ELM-rbf method has poor performance which is probably due to the insufficient hidden layer nodes. Hence, the labeled samples in semisupervised methods are not the only reason that results in a higher accuracy. It can also be seen that WDTELM is better than DAELM in terms of average classification performance. Therefore, WDTELM is considered to be more effective than other tested methods in general.

For online learning process, we wish to achieve approximately the same if not better accuracies of their batch learning versions. To evaluate the overall performance, we conducted experiments on both DAELM and WDTELM. For comparison purpose, we use SSA and CSS on DAELM and OWDTELM, respectively, so as to see the difference between them and their batch versions. Table 5 shows the results on the online version of DAELM. It can be noted that the performance of the online version reaches similar classification accuracy as DAELM does. For some batches, it surpasses its batch learning method, for example, batches (3) and (10). The reason may be because the online learning process updates the model whenever a new sample arrives, and this procedure may better help learn the difference between samples.

Table 6 shows the classification accuracy of OWDTELM. The accuracies were recorded after the method updates all the unlabeled samples in each batch. As shown in the table, the accuracy of OWDTELM remains the same as its batch learning version does. Although little disturbance occurs, for example, batches (2) and (3) when $k = 8$, the gap is not significant. For batches (2) and (3), it has less than 2% accuracy decrease while for batch (10) it increases by 3%. It can be noted that the average accuracy for OWDTELM is higher than that of WDTELM for some values of k such as $k = 6$. The difference is below 1.5% which can be partly due to the experimental error caused by randomization of the ELM neurons. In general, the OWDTELM achieves similar performance as its batch learning version does.

The processing time was recorded whenever ODAELM and OWDTELM updated the model. On average, the processing time of ODAELM takes 0.0948 seconds and fluctuates within 0.0139. OWDTELM takes slightly more time, that is, 0.0958 on average, and fluctuates within 0.0125. This is different from batch learning whose processing time will increase with the size of the samples. We also replace the online learning updating with their batch learning, that is, DAELM and WDTELM, respectively. In this case, the methods will retrain the model from scratch whenever the new sample arrives. The processing time increases drastically. With 400 unlabeled samples, the updating procedure took over 0.2 seconds for both methods, which is over 2 times of its online version. If the size of the samples grows too large, the updating will take minutes, and even hours, to complete.

5. Discussion

In the evaluation of the proposed methods in the paper, we compared their performances regarding the classification accuracy. In general, the proposed CSS helps in choosing more representative samples as KS in DAELM does, and the weighted strategy further helps the learning process to be more accurate. The online versions of DAELM and WDTELM can achieve similar performance as their batch learning versions do and are suitable for scenarios where unlabeled samples are not accessible before training.

It should also be noted that the proposed methods in this paper are not limited to E-Nose systems only. Other domains that have similar characteristics are also worth trying. Meanwhile, the online version of the proposed method only considered a simple scenario as a head start. More sophisticated cases including, but not limited to, labeled sample incremental learning, samples switched from unlabeled to labeled, and data distribution changes should be considered for future researches.

6. Conclusions

In this paper, we proposed WDTELM to reduce the number of labeled samples in DAELM while achieving similar or better performance. Then, aiming at online learning process, we proposed online learning versions of DAELM and WDTELM, named ODTELM and OWDTELM, respectively, which allow the unlabeled samples to be added to the classifier in a one-by-one or chunk-by-chunk manner after labeled samples are given. The experimental results show that WDTELM outperforms DAELM with less labeled samples regarding the classification accuracy. Meanwhile, it also possesses higher performance than other commonly used approaches such as SVM, ELM, and ensemble based methods. The experiments for online versions of DAELM and WDTELM have verified that they each possess the same classification accuracy as their batch learning version does. The processing time also confirms that the online versions are time-saving methods compared with their batch learning versions.

Abbreviations

KS:	Kennard and Stone method
CSS:	Clustering-Aided Sample Selection
SVM:	Support Vector Machine
ELM:	Extreme Learning Machine
DAELM:	Domain Adaptation Extreme Learning Machine
WDTELM:	Weighted Domain Transfer Extreme Learning Machine
ODTELM:	Online Domain Transfer Extreme Learning Machine.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the Ministry of Science and Technology Department Foundation of Sichuan Province, under Grants nos. 2016GZ0075 and 2016GZ0077, and partially supported by the Science and Technology Innovation Seed Project of Sichuan Province under Grant no. 2017RZ0008.

The work of N. Wang was supported by the National Natural Science Foundation of China (NSFC) under Grant 61604054. The authors would like to thank Pei Yang from South China University of Technology and Zhipeng Cai from Georgia State University for their suggestions on refining the ideas in the paper. Additionally, the authors would like to thank Lei Zhang from Chongqing University for his instructions on implementing DAELM.

References

[1] Z. Cai, G. Lin, and G. Xue, "Improved approximation algorithms for the capacitated multicast routing problem," in *Computing and combinatorics*, vol. 3595 of *Lecture Notes in Computer Science*, pp. 136–145, Springer, 2005.

[2] Z. Cai, Z.-Z. Chen, and G. Lin, "A 4713-approximation algorithm for the capacitated multicast tree routing problem," *Theoretical Computer Science*, vol. 410, no. 52, pp. 5415–5424, 2009.

[3] Z. Cai, R. Goebel, and G. Lin, "Size-constrained tree partitioning: approximating the multicast k-tree routing problem," *Theoretical Computer Science*, vol. 412, no. 3, pp. 240–245, 2011.

[4] L. Jin, H. Xian, J. Bie, Y. Sun, H. Hou, and Q. Niu, "License plate recognition algorithm for passenger cars in Chinese residential areas," *Sensors*, vol. 12, no. 6, pp. 8355–8370, 2012.

[5] X. Xi, G. Yang, Y. Yin, and X. Meng, "Finger vein recognition with personalized feature selection," *Sensors*, vol. 13, no. 9, pp. 11243–11259, 2013.

[6] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional Bi-directional LSTM networks," *Sensors*, vol. 17, no. 2, article no. 273, 2017.

[7] A. Ponzoni, C. Baratto, N. Cattabiani et al., "Smetal oxide gas sensors, a survey of selectivity issues addressed at the SENSOR lab, Brescia (Italy)," *Sensors*, vol. 17, no. 4, article no. 714, 2017.

[8] M. Holmberg and T. Artursson, *Drift Compensation, Standards, and Calibration Methods*, WILEY-VCH: Weinheim, Germany, 1st edition, 2004.

[9] A. Hierlemann and R. Gutierrez-Osuna, "Higher-order chemical sensing," *Chemical Reviews*, vol. 108, no. 2, pp. 563–613, 2008.

[10] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," *Sensors and Actuators B: Chemical*, vol. 166-167, pp. 320–329, 2012.

[11] I. Rodriguez-Lujan, J. Fonollosa, A. Vergara, M. Homer, and R. Huerta, "On the calibration of sensor arrays for pattern recognition using the minimal number of experiments," *Chemometrics and Intelligent Laboratory Systems*, vol. 130, pp. 123–134, 2014.

[12] H. J. S. Dorren and R. K. Snieder, "Methodology for the selection of suitable sensors for incorporation into a gas sensor array," *Analytica Chimica Acta*, vol. 242, pp. 31–36, 1991.

[13] T. T. Sunil, S. Chaudhuri, and V. Mishra, "Optimal selection of SAW sensors for E-Nose applications," *Sensors and Actuators B: Chemical*, vol. 219, pp. 238–244, 2015.

[14] P. T. Moseley, "Materials selection for semiconductor gas sensors," *Sensors and Actuators B: Chemical*, vol. 6, no. 1-3, pp. 149–156, 1992.

[15] R. G. Pavelko, A. A. Vasiliev, E. Llobet et al., "Comparative study of nanocrystalline SnO2 materials for gas sensor application: Thermal stability and catalytic activity," *Sensors and Actuators B: Chemical*, vol. 137, no. 2, pp. 637–643, 2009.

[16] B.-C. Serban, M. Brezeanu, C. Cobianu et al., "Materials selection for gas sensing. An HSAB perspective," in *Proceedings of the 37th International Semiconductor Conference, CAS 2014*, pp. 21–30, Romania, October 2014.

[17] A. Marikutsa, M. Rumyantseva, A. Baranchikov, and A. Gaskov, "Nanocrystalline BaSnO3 as an alternative gas sensor material: Surface reactivity and high sensitivity to SO2," *Materials* , vol. 8, no. 9, pp. 6437–6454, 2015.

[18] J. Fonollosa, I. Rodríguez-Luján, and R. Huerta, "Chemical gas sensor array dataset," *Data in Brief*, vol. 3, pp. 85–89, 2015.

[19] H. Liu and Z. Tang, "Metal oxide gas sensor drift compensation using a dynamic classifier ensemble based on fitting.," *Sensors*, vol. 13, no. 7, pp. 9160–9173, 2013.

[20] H. Liu, R. Chu, and Z. Tang, "Metal oxide gas sensor drift compensation using a two-dimensional classifier ensemble," *Sensors*, vol. 15, no. 5, pp. 10180–10193, 2015.

[21] L. Zhang and D. Zhang, "Domain adaptation extreme learning machines for drift compensation in E-nose systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 7, pp. 1790–1801, 2015.

[22] J. Fonollosa, L. Fernández, A. Gutiérrez-Gálvez, R. Huerta, and S. Marco, "Calibration transfer and drift counteraction in chemical sensor arrays using Direct Standardization," *Sensors and Actuators B: Chemical*, vol. 236, pp. 1044–1053, 2016.

[23] J.-W. Gong, Q.-F. Chen, M.-R. Lian, N.-C. Liu, and C. Daoust, "Temperature feedback control for improving the stability of a Semiconductor-Metal-Oxide (SMO) gas sensor," *IEEE Sensors Journal*, vol. 6, no. 1, pp. 139–145, 2006.

[24] J. Rebholz, U. Weimar, and N. Barsan, "Influence of conduction mechanism changes on the sensor performance of SMOX based gas sensors," in *Proceedings of the 28th European Conference on Solid-State Transducers, EUROSENSORS 2014*, pp. 20–23, Italy, September 2014.

[25] A. Ziyatdinov, A. Chaudry, K. Persaud, P. Caminal, and A. Perera, "Common principal component analysis for drift compensation of gas sensor array data," in *Proceedings of the 13th International Symposium on Olfaction and Electronic Nose, ISOEN*, pp. 566–569, Italy, April 2009.

[26] M. Padilla, A. Perera, I. Montoliu, A. Chaudry, K. Persaud, and S. Marco, "Drift compensation of gas sensor array data by Orthogonal Signal Correction," *Chemometrics and Intelligent Laboratory Systems*, vol. 100, no. 1, pp. 28–35, 2010.

[27] T. Artursson, T. Eklöv, I. Lundström, P. Mårtensson, M. Sjöström, and M. Holmberg, "Drift correction for gas sensors using multivariate methods," *Journal of Chemometrics*, vol. 14, no. 5-6, pp. 711–723, 2000.

[28] D. Hui, L. Jun-Hua, and S. Zhong-Ru, "Drift reduction of gas sensor by wavelet and principal component analysis," *Sensors and Actuators B: Chemical*, vol. 96, no. 1-2, pp. 354–363, 2003.

[29] K. Yan and D. Zhang, "Calibration transfer and drift compensation of e-noses via coupled task learning," *Sensors and Actuators B: Chemical*, vol. 225, pp. 288–297, 2016.

[30] S. Di Carlo, M. Falasconi, E. Sanchez, A. Scionti, G. Squillero, and A. Tonda, "Increasing pattern recognition accuracy for chemical sensing by evolutionary based drift compensation," *Pattern Recognition Letters*, vol. 32, no. 13, pp. 1594–1603, 2011.

[31] J. Fonollosa, A. Vergara, and R. Huerta, "Algorithmic mitigation of sensor failure: Is sensor replacement really necessary?" *Sensors and Actuators B: Chemical*, vol. 183, pp. 211–221, 2013.

[32] S.-F. Ding, X.-Z. Xu, and R. Nie, "Extreme learning machine and its applications," *Neural Computing and Applications*, vol. 25, no. 3, pp. 549–556, 2014.

[33] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, 2013.

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

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

[36] H. Zou, X. Lu, H. Jiang, and L. Xie, "A fast and precise indoor localization algorithm based on an online sequential extreme learning machine," *Sensors*, vol. 15, no. 1, pp. 1804–1824, 2015.

[37] 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.

[38] W.-Y. Deng, Y.-S. Ong, and Q.-H. Zheng, "A Fast Reduced Kernel Extreme Learning Machine," *Neural Networks*, vol. 76, pp. 29–38, 2016.

[39] Z. Zhao, Z. Chen, Y. Chen, S. Wang, and H. Wang, "A class incremental extreme learning machine for activity recognition," *Cognitive Computation*, vol. 6, no. 3, pp. 423–431, 2014.

[40] H. Kaya, A. A. Karpov, and A. A. Salah, "Robust acoustic emotion recognition based on cascaded normalization and extreme learning machines," *International Symposium on Neural Networks*, pp. 115–123, 2016.

[41] P. Mohapatra, S. Chakravarty, and P. K. Dash, "An improved cuckoo search based extreme learning machine for medical data classification," *Swarm and Evolutionary Computation*, vol. 24, pp. 25–49, 2015.

[42] A. Iosifidis, A. Tefas, and I. Pitas, "Approximate kernel extreme learning machine for large scale data classification," *Neurocomputing*, vol. 219, pp. 210–220, 2017.

[43] J. Xin, Z. Wang, L. Qu, G. Yu, and Y. Kang, "A-ELM: adaptive distributed extreme learning machine with MapReduce," *Neurocomputing*, vol. 174, pp. 368–374, 2015.

[44] B. Wang, S. Huang, J. Qiu, Y. Liu, and G. Wang, "Parallel online sequential extreme learning machine based on MapReduce," *Neurocomputing*, vol. 149, pp. 224–232, 2015.

[45] A. Vergara, Gas sensor array drift dataset at different concentrations data set. http://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations/ accessed on October 23, 2013.

[46] S. Cheng, Z. Cai, and J. Li, "Curve query processing in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, 1 page, 2014.

[47] Z. He, Z. Cai, S. Cheng, and X. Wang, "Approximate aggregation for tracking quantiles and range countings in wireless sensor networks," *Theoretical Computer Science*, vol. 607, part 3, pp. 381–390, 2015.

[48] J. Li, S. Cheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Approximate holistic aggregation in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 13, no. 2, article no. 11, 2017.

[49] I. T. Jolliffe, *Principal Component Analysis*, vol. 87, Springer, New York, NY, USA, 1986.

[50] S. Cheng, Z. Cai, J. Li, and X. Fang, "Drawing dominant dataset from big sensory data in wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 531–539, Hong Kong, April 2015.

[51] S. Cheng, Z. Cai, J. Li, and H. Gao, "Extracting kernel dataset from big sensory data in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 4, pp. 813–827, 2017.

[52] Y. Wang, F. Cao, and Y. Yuan, "A study on effectiveness of extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2483–2490, 2011.

[53] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2009.