Information Sciences 503 (2019) 635–651

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An off-center technique: Learning a feature transformation to improve the performance of clustering and classification

Dasen Yan[a], Xinlei Zhou[a], Xizhao Wang[a,d], Ran Wang[b,c,*]

[a] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[b] College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, China
[c] Shenzhen Key Laboratory of Advanced Machine Learning and Applications, Shenzhen University, Shenzhen 518060, China
[d] The Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Article history:
Received 22 March 2019
Revised 11 June 2019
Accepted 30 June 2019
Available online 11 July 2019

Keywords:
Feature transformation
Feed-forward neural network
Similarity matrix

ABSTRACT

This paper proposes a feature transformation method to improve the performance of clustering and classification, which is named as weight-matrix learning (WML). A feed-forward neural network is particularly designed for WML, which aims to learn the optimal weights by minimizing an objective function similar to cross-entropy, and the training process is finished based on the technique of batch gradient descent or stochastic gradient descent. The proposed feature transformation is linear, which is a non-trivial extension of a previous technique named feature-weight learning (FWL). Essentially, WML can be considered as a learning technique of departing 0.5-similarity, since it can make the samples with similarity larger than 0.5 closer and the samples with similarity lower than 0.5 farther away. From this perspective, WML is identified as an off-center technique with the center of 0.5-similarity. Theoretically and experimentally, it is validated that WML can significantly improve the performance of some clustering algorithms like *k*-means, and enhance the performance of some classification algorithms like random weight neural network.

1. Introduction

Feature transformation is critical for machine learning tasks such as clustering, classification, regression, and semi-supervised learning, etc. For a clustering (classification) task, if the similarity of samples is made large within a cluster (class) and made small among different clusters (classes) through a feature transformation, the difficulty of the clustering (classification) task will be possibly reduced. A similarity measure is usually associated with a distance metric. In order to get a better distance metric for the transformed data, learning techniques have been proposed and extensively studied. A better metric has led to an improvement of performance in both supervised and unsupervised learning [27].

Learning a good distance metric in feature space is a key issue in many models. The goal of distance metric learning is to learn a transformation matrix, in order to transform the samples from the original space to a new feature space such that the similarity of samples can be measured in a better way. In the past decade, researchers have carried out a lot of studies on distance metric learning [4,21,23,28,29]. According to the availability of labels in the training set, the algorithms can be divided into two categories, i.e., supervised and unsupervised distance metric learning. The main idea of supervised distance metric learning is using the prior knowledge of training samples, through optimizing a certain objective function,

* Corresponding author.
E-mail addresses: ranwang3-c@my.cityu.edu.hk, wangran@szu.edu.cn (R. Wang).

https://doi.org/10.1016/j.ins.2019.06.068
0020-0255/

to obtain a metric matrix such that data points from the same class are put closely together whereas those from different classes are moved far apart. Unsupervised distance metric learning is a more challenging task due to the lack of labels. Most of them can be used to mine the underlying manifold structure of data, e.g., learning a basic low-dimensional manifold that preserves the geometric relationships (such as distance) among the observed samples.

Many metric-based learning methods have been extensively studied, e.g., Yeung and Wang et al. [24,30] proposed a feature-weight learning (FWL) method to acquire a weight vector for the input features. It helps getting a better similarity measure for weighted data, and is used to improve the performance of clustering algorithms. FWL is an unsupervised method that has been successfully applied to clustering algorithms like fuzzy c-means. The main idea is to use the information of training samples, through minimizing the fuzziness of the similarity matrix, to obtain a weight vector for the features. By weighting each feature, it gets a weighted Euclidean distance for each pair of samples, as a result, samples in the same group will have higher similarities and samples of different groups will have lower similarities. In FWL, the weighting technique is a mature processing technology that has been successfully applied to various fields such as fuzzy soft multiset [5], decision making [12,13], and so on. However, FWL assigns each feature a single weight without considering the relations among different features. In order to solve this problem, in this paper, we will propose a new method named weight-matrix learning (WML). The contributions of this article are summarized as follows.

- We propose the WML method, which aims to learn a matrix that transforms the data from the original space to a new feature space. The data in the new feature space will have a better representation for clustering and classification, i.e., samples with similarity larger than 0.5 will be closer and samples with similarity smaller than 0.5 will be farther away. From this perspective, WML is identified as an off-center technique with the center of 0.5-similarity. As a result, it can help reduce the difficulty of the learning task.
- WML is a non-trivial extension of FWL. FWL assigns each feature a single weight without considering the relations among different features. WML extends the weight vector in FWL to an ordinary square matrix with full rank. Since the non-diagonal elements in the matrix are generally non-zero, it can take advantage of the inter-correlations among features. In addition, extending the weight vector to a square matrix enlarges the modifiability of the transformation, thereby enhancing the data transformation capability.
- The similarity measurement in WML is based on a pseudo-distance, i.e., the square of weighted distance, rather than the distance itself. This improvement reduces the complexity for computing the similarity matrix to some extent.
- We place the WML into a feed-forward neural network (FFNN). The network structure gives a good explanation for WML. Moreover, classical techniques like stochastic gradient descent, batch gradient descent, and other gradient-based algorithms can be used for training.
- We demonstrate theoretically and experimentally that WML can significantly reduce the uncertainty of similarity matrix thereby improve the performance of clustering and classification.

The remainder of this paper is organized as follows. Section 2 proposes the WML method for feature transformation. In Section 3, the advantages of the proposed WML method compared with the existing FWL method are discussed. Section 4 experimentally compares and analyzes the impact of FWL and WML on clustering and classification tasks. Finally, Section 5 concludes the paper.

2. The proposed WML method

It is demonstrated in [30] that in a learning problem, uncertainty is inevitable when making decision for a sample. This uncertainty is usually caused by the fuzziness of the similarity matrix. In this section, we will propose the WML method, which essentially performs a feature transformation to reduce the uncertainty of similarity matrix, thereby reduces the difficulty of the learning task. The proposed WML method is unsupervised, but the transformed data can be used in both supervised and unsupervised learning. Thus, WML can be treated as a data pre-processing technique.

2.1. Feature transformation

Suppose that $\mathbb{S} \subset \mathcal{R}^n$ is a data set containing N column vectors, which is represented as

$$\mathbb{S} = \{\vec{x}_i | \vec{x}_i \in \mathcal{R}^n, i = 1, 2, \cdots, N\}. \tag{1}$$

$\mathbb{S}$ can be converted into a new data set by a transformation matrix W, i.e.,

$$\mathbb{S}_W = \{\vec{y}_i | \vec{y}_i = W\vec{x}_i, i = 1, 2, \cdots, N\}. \tag{2}$$

Assume that W is a $\tilde{n} \times n$ matrix, thus

1. when $\tilde{n} > n$, $W\vec{x}_i$ increases the number of features for each sample;
2. when $\tilde{n} < n$, $W\vec{x}_i$ performs a dimensionality reduction for each sample;
3. when $\tilde{n} = n$, $W\vec{x}_i$ is essentially a feature transformation process that converts the n original features to n new features.

In this paper, we only consider the third case, i.e., $\tilde{n} = n$. Thus, $W = [w_{ij}]_{n\times n}$ can be treated as a weight matrix with full rank, and how to learn the optimal W will be the key issue throughout the paper.

The weight matrix W will affect the distance measures of samples. The Euclidean distance between two vectors is defined as

$$d^2(\vec{x}_p, \vec{x}_q) = \sum_{j=1}^{n} (x_{jp} - x_{jq})^2, \tag{3}$$

where

$$\begin{cases} \vec{x}_p = [x_{1p}, x_{2p}, \cdots, x_{np}]^\top \\ \vec{x}_q = [x_{1q}, x_{2q}, \cdots, x_{nq}]^\top \end{cases} \tag{4}$$

Noting that $d^2(\vec{x}_p, \vec{x}_q)$ can also be expressed as

$$d^2(\vec{x}_p, \vec{x}_q) = \| \vec{x}_p - \vec{x}_q \|^2 = (\vec{x}_p - \vec{x}_q)^\top (\vec{x}_p - \vec{x}_q), \tag{5}$$

and we have

$$d^2(\vec{y}_p, \vec{y}_q) = \| \vec{y}_p - \vec{y}_q \|^2 = (\vec{y}_p - \vec{y}_q)^\top (\vec{y}_p - \vec{y}_q) = (W(\vec{x}_p - \vec{x}_q))^\top (W(\vec{x}_p - \vec{x}_q)) = (\vec{x}_p - \vec{x}_q)^\top (W^\top W)(\vec{x}_p - \vec{x}_q). \tag{6}$$

Since W is a full rank matrix, $W^\top W$ is an orthogonal matrix. It is worth noting that, the Euclidean distance between the transformed vectors $\vec{y}_p$ and $\vec{y}_q$ is the Mahalanobis distance between the original vectors $\vec{x}_p$ and $\vec{x}_q$.

2.2. Uncertainty and fuzziness of similarity matrix

Similarity of samples is usually measured by a distance metric. For example, Euclidean distance was used in [26] to measure the similarity between data points. Different from this work, in this paper, Mahalanobis distance is applied. The similarity between $\vec{x}_p$ and $\vec{x}_q$ under the transformation of matrix W is defined by the following equation

$$\rho_{pq}^{(W)} = \frac{1}{1 + \beta \cdot d_{pq}^{(W)}}. \tag{7}$$

In Eq. (7), $d_{pq}^{(W)}$ is defined as

$$d_{pq}^{(W)} = d^2(\vec{y}_p, \vec{y}_q) = (\vec{x}_p - \vec{x}_q)^\top (W^\top W)(\vec{x}_p - \vec{x}_q), \tag{8}$$

β is a positive number that can be calculated by solving Eq. (9),

$$\frac{2}{N(N-1)} \sum_{q>p} \rho_{pq}^{(I)} = 0.5, \tag{9}$$

where N is the number of samples, $\rho_{pq}^{(I)}$ is the value of $\rho_{pq}^{(W)}$ at $W = I$, indicating the similarity between the original points $\vec{x}_p$ and $\vec{x}_q$ without transformation. Intuitively, parameter β is to balance the data distribution in order to have an average sample similarity around 0.5. Then, the similarity matrix of samples in $\mathbb{S}$ under the transformation of matrix W is formulated as

$$\rho_{\mathbb{S}}^{(W)} = \begin{bmatrix} \rho_{11}^{(W)} & \rho_{12}^{(W)} & \cdots & \rho_{1N}^{(W)} \\ \rho_{21}^{(W)} & \rho_{22}^{(W)} & \cdots & \rho_{2N}^{(W)} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N1}^{(W)} & \rho_{N2}^{(W)} & \cdots & \rho_{NN}^{(W)} \end{bmatrix}. \tag{10}$$

According to Basak et al. [1], the uncertainty of this similarity matrix could be reduced by minimizing the following objective function:

$$E(W) = \frac{1}{N(N-1)} \sum_{q<p} E_{pq}(W) = \frac{1}{N(N-1)} \sum_{q<p} \left[\rho_{pq}^{(W)} (1 - \rho_{pq}^{(I)}) + \rho_{pq}^{(I)} (1 - \rho_{pq}^{(W)}) \right], \tag{11}$$

where N is the number of samples, W is the feature transformation matrix, $\rho_{pq}^{(W)}$ is defined in Eq. (7), and $\rho_{pq}^{(I)}$ is mentioned in Eq. (9).

The objective function $E(W)$ is derived from a simple function, i.e.,

$$f(x, y) = y(1 - x) + x(1 - y), \text{ where } 0 \le x, y \le 1. \tag{12}$$

From this function, we have $\frac{\partial f}{\partial x} = 1 - 2y$, thus

$$\begin{cases} \dfrac{\partial f}{\partial x} > 0, & \text{when } y < 0.5 \\ \dfrac{\partial f}{\partial x} < 0, & \text{when } y > 0.5 \end{cases} \tag{13}$$

Therefore, from Eq. (11), we know that $E(W)$ with respect to $\rho_{pq}^{(W)}$ is a strictly monotonically increasing function under the condition of fixed $\rho_{pq}^{(I)} < 0.5$ and is a strictly monotonically decreasing function under the condition of fixed $\rho_{pq}^{(I)} > 0.5$. Based on this statement, we have the following equality:

$$\mathrm{Lim}_{[\rho_{pq}^{(W)}\to 0,\ \rho_{pq}^{(I)}<0.5] \text{ or } [\rho_{pq}^{(W)}\to 1,\ \rho_{pq}^{(I)}>0.5]} E_{pq}(W) = \min E_{pq}(W). \tag{14}$$

Thus, minimizing Eq. (11) implies that the new similarity of transformed samples by matrix W tends to be 1 (0) if the old similarity of original samples is greater (less) than 0.5. In a learning problem, a consensus is that samples within the same cluster (class) tend to have higher similarities, i.e., >0.5, and samples from different clusters (classes) tend to have lower similarities, i.e., <0.5. Therefore, under the influence of matrix W, the average sample similarity within the same cluster (class) will increase and the average sample similarity among different clusters (classes) will decrease.

Moreover, according to [1], the fuzziness of the similarity matrix $\rho_{\mathbb{S}}^{(W)}$ can be defined as

$$\mathrm{Fuzziness}(\rho_{\mathbb{S}}^{(W)}) = -\tfrac{1}{N(N-1)} \sum_{q<p} \left[\rho_{pq}^{(W)} \log \rho_{pq}^{(W)} + (1-\rho_{pq}^{(W)}) \log(1-\rho_{pq}^{(W)})\right]. \tag{15}$$

As we can see from Eq. (15), the farther the value of $\rho_{pq}^{(W)}$ from 0.5, the smaller the fuzziness value is. The values in the similarity matrix can be made away from 0.5 by minimizing the objective function Eq. (11). So the fuzziness of the similarity matrix will decrease as $E(W)$ decreases.

2.3. Algorithm description

We use the gradient descent technique to learn the feature transformation matrix W by minimizing Eq. (11). The change in W, denoted as ΔW, is defined as

$$\Delta W = -\eta \frac{\partial E(W)}{\partial W} \tag{16}$$

where η is the learning rate. Specifically, $\frac{\partial E(W)}{\partial W}$ is defined as

$$\frac{\partial E(W)}{\partial W} \triangleq \begin{bmatrix} \frac{\partial E(W)}{\partial w_{11}} & \frac{\partial E(W)}{\partial w_{12}} & \cdots & \frac{\partial E(W)}{\partial w_{1n}} \\ \frac{\partial E(W)}{\partial w_{21}} & \frac{\partial E(W)}{\partial w_{22}} & \cdots & \frac{\partial E(W)}{\partial w_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E(W)}{\partial w_{n1}} & \frac{\partial E(W)}{\partial w_{n2}} & \cdots & \frac{\partial E(W)}{\partial w_{nn}} \end{bmatrix}. \tag{17}$$

In computing $\frac{\partial E(W)}{\partial W}$, the following expression is used

$$\frac{\partial E(W)}{\partial W} = \frac{1}{N(N-1)} \sum_{q<p} (1 - 2\cdot\rho_{pq}^{(I)}) \cdot \frac{\partial \rho_{pq}^{(W)}}{\partial d_{pq}^{(W)}} \cdot \frac{\partial d_{pq}^{(W)}}{\partial W}, \tag{18}$$

where

$$\begin{cases} \frac{\partial \rho_{pq}^{(W)}}{\partial d_{pq}^{(W)}} = \frac{-\beta}{(1+\beta\cdot d_{pq}^{(W)})^2} \\ \frac{\partial d_{pq}^{(W)}}{\partial W} = 2(\vec{x}_p - \vec{x}_q)(\vec{x}_p - \vec{x}_q)^\top W^\top \end{cases} \tag{19}$$

The learning rate η for each epoch can be given empirically or determined by

$$E(W - \eta \frac{\partial E(W)}{\partial W}) = \min_{\lambda>0} E\left(W - \lambda \frac{\partial E(W)}{\partial W}\right). \tag{20}$$

Generally, Eq. (20) is designed to get a suitable learning step-length for the next epoch. Calculating a suitable step-length for each epoch instead of using a fixed step-length may speed up the training process.

Finally, the training process for W is described in Algorithm 1. After training, we can get a well-learned transformation matrix W and the objective function $E(W)$ will reach a local minimum. That is, the fuzziness of similarity matrix with transformation matrix W will be much smaller than the fuzziness of similarity matrix without transformation.

In is noteworthy that σ in Algorithm 1 is an early stopping threshold, the training will be stopped before the maximum number of epoches if the decrease of the loss value in the current iteration compared with the previous iteration is less than σ. Usually, the decrease of the loss value becomes less and less as the number of epoches increases. Thus, the learning could be stopped earlier with a larger σ and later with a smaller σ. Based on our experience, a large value of σ may cause

Algorithm 1: WML algorithm.

Input: Data set $\mathbb{S} = \{\vec{x}_i | \vec{x}_i \subset \mathcal{R}^n, i = 1, 2, \cdots, N\}$; maximum number of epoches M, early stopping threshold σ.
Output: Transformation matrix W.

```
Initialize W = I, epo=0;
Calculate β by solving Equation (9) for Equation (7);
Calculate E_0(W) based on Equation (11);
while epo + 1 ≤ M do
    Calculate ∂E(W)/∂W according to Equation (18);
    Get the learning rate η by solving Equation (20);
    Update W = W + ΔW, epo = epo + 1;
    Calculate E_epo(W) based on Equation (11);
    if |E_epo(W) − E_epo−1(W)| < σ then
        break;
    end
end
return Transformation matrix W.
```

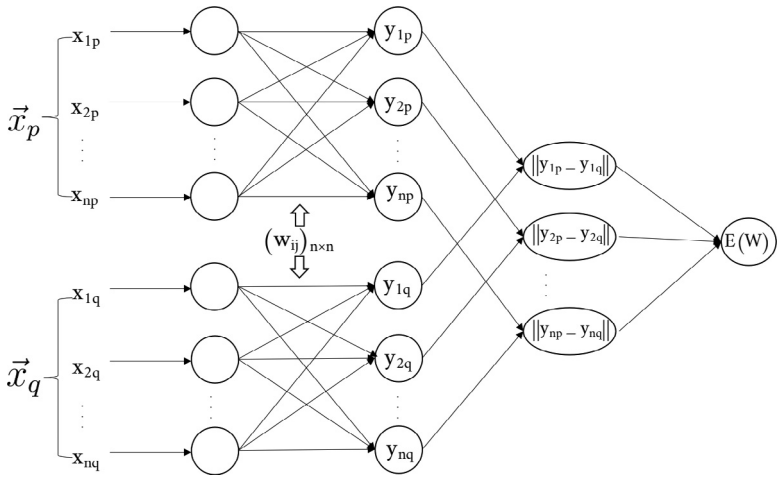


Fig. 1. Network representation for WML.

the learning process to terminate prematurely, resulting in bad performance. Thus, σ is set as a very small number, i.e., 0.001.

3. Advantages of WML

Intuitively, the proposed WML can be represented as a FFNN as shown in Fig. 1. The input of the network is a pair of original data, the output is the objective function $E(W)$ to be minimized, the weight between the first layer and the second layer is the transformation matrix W which is shared by the upper and lower connections. This neural network model can be trained by the batch gradient descent algorithm or the stochastic gradient descent algorithm. Our aim in this phase is to get a well-learned matrix W for data transformation through training the network.

It is noteworthy that when the transformation matrix W is restricted to be a diagonal matrix, the proposed WML method degenerates back to the previous FWL method in [24,30], which has been successfully applied to improve clustering performance. In other words, WML is an improvement and extension of FWL. Compared with FWL, WML has the following advantages.

3.1. Stronger feature transformation capability

The parameter space of FWL is

$$\left\{ W = \begin{bmatrix} w_{11} & 0 & \dots & 0 \\ 0 & w_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_{nn} \end{bmatrix} \middle| w_{ii} \neq 0, i = 1, \dots, n \right\}, \tag{21}$$

and the parameter space of the proposed WML is

$$\left\{ W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{bmatrix} \middle| W \in \mathcal{R}^{n\times n} \text{ is a full rank matrix} \right\}. \tag{22}$$

Obviously, the parameter space of FWL is a true subset of the parameter space of WML, both FWL and WML have the same objective function, i.e., Eq. (11).

Given an original sample $\vec{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^\top$, traditional FWL will transform it to a new sample $\vec{y}_i = [y_{i1}, y_{i2}, \dots, y_{in}]^\top$, where $y_{ij} = x_{ij} \times w_{jj}, j = 1, \dots, n$. In this case, each attribute of $\vec{y}_i$ is completely and independently decided by the corresponding attribute of $\vec{x}_i$.

However, the proposed WML will transform $\vec{x}_i$ to another sample $\vec{y}_i = [y_{i1}, y_{i2}, \dots, y_{in}]^\top$, where $y_{ij} = \sum_{k=1}^{n} x_{ik} \times w_{jk}, j = 1, \dots, n$. Since the non-diagonal elements in the transformation matrix W are generally non-zero, each attribute of $\vec{y}_i$ is a weighted sum of all the attributes in $\vec{x}_i$. That means, each new feature is a linear combination of all the original features, and the weights of the original features are learned by Algorithm 1 from the training data. From this perspective, we can say that WML takes advantage of the inter-correlations among attributes.

In order to intuitively observe the feature transformation capability of FWL and WML, we take data set *Iris* as an example, and draw the data distributions before and after transformation. For convenience, the first and second features in *Iris* are selected for visualization. Fig. 2 demonstrates the distributions of original data, data transformed by FWL and data transformed by WML. Obviously, both FWL and WML can affect the data distribution, but in Fig. 2(c), the data transformed by WML has the smallest within-class sample distance and the largest between-class sample distance. By comparing Fig. 2(a) and (b), it is observed that FWL only realizes a scaling for each feature. While by comparing Fig. 2(a) and (c), it can be investigated that WML makes the data of the same category more compact and the data of different categories sparser. Thus, WML possesses a stronger feature transformation capability than FWL.

3.2. Less uncertainty of similarity matrix

WML can get a similarity matrix with less uncertainty in comparison with FWL. Given the similarity matrix of original data, the objective function used by both WML and FWL, i.e., Eq. (11), is to decrease the values of the elements smaller than 0.5, and to increase the values of the elements larger than 0.5. In the following, we make an intuitive illustration. Suppose we have a randomly generated data set, i.e.,

$$\begin{aligned} \mathbb{S} = \ & \{(0.420, 0.750, 0.640, 0.200),\ (0.880, 0.300, 0.550, 0.900),\ (0.774, 0.467, 0.877, 0.660), \\ & (0.241, 0.259, 0.822, 0.885)\}. \end{aligned} \tag{23}$$

Based on Eq. (7), we can compute the original similarity matrix $\rho_{\mathbb{S}}^{(I)}$, i.e.,

$$\rho_{\mathbb{S}}^{(I)} = \begin{bmatrix} 1.000 & 0.428 & 0.509 & 0.448 \\ 0.428 & 1.000 & 0.612 & 0.506 \\ 0.509 & 0.612 & 1.000 & 0.536 \\ 0.448 & 0.506 & 0.536 & 1.000 \end{bmatrix}, \tag{24}$$

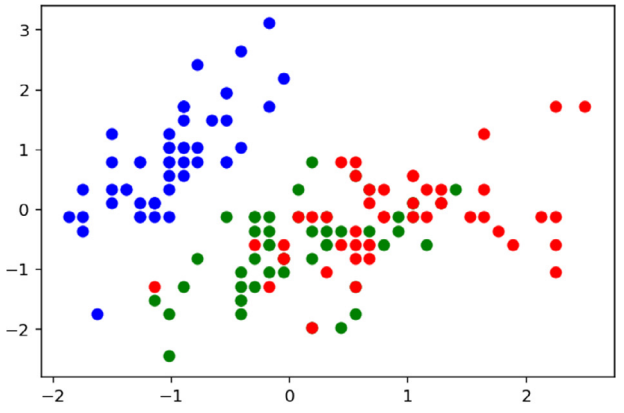
where the balance parameter β is calculated as 0.602. After training by FWL and WML respectively, we get two new similarity matrices, i.e.,

$$\rho_{\mathbb{S}}^{(FWL)} = \begin{bmatrix} 1.000 & 0.302 & 0.397 & 0.304 \\ 0.302 & 1.000 & 0.556 & 0.776 \\ 0.397 & 0.556 & 1.000 & 0.552 \\ 0.304 & 0.776 & 0.552 & 1.000 \end{bmatrix}, \tag{25}$$

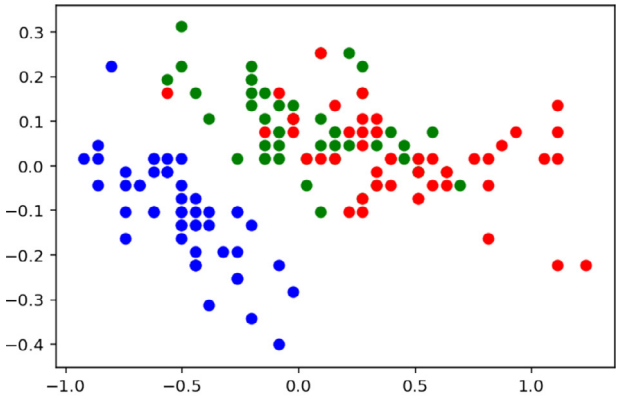
and

$$\rho_{\mathbb{S}}^{(WML)} = \begin{bmatrix} 1.000 & 0.248 & 0.436 & 0.251 \\ 0.248 & 1.000 & 0.734 & 0.996 \\ 0.436 & 0.734 & 1.000 & 0.741 \\ 0.251 & 0.996 & 0.741 & 1.000 \end{bmatrix}. \tag{26}$$

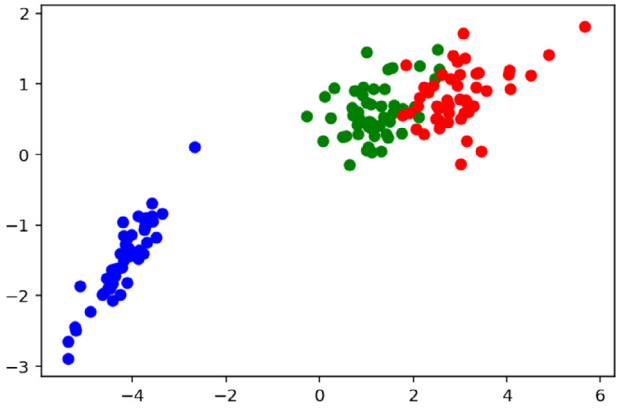
It can be seen that in matrix $\rho_{\mathbb{S}}^{(I)}$, the elements smaller than 0.5 include (0.428, 0.448) and the elements larger than 0.5 include (0.509, 0.612, 0.506, 0.536). After FWL training, the corresponding elements become (0.302, 0.304) and (0.397, 0.556, 0.776, 0.552), respectively. After WML training, the corresponding elements become (0.248, 0.251) and (0.436, 0.734, 0.996, 0.741), respectively. Obviously, the values in $\rho_{\mathbb{S}}^{(WML)}$ is farther away from 0.5 than those in $\rho_{\mathbb{S}}^{(FML)}$. That is, the proposed WML method obtains a similarity matrix with less uncertainty.



(a) Data distribution in *Iris* before transformation



(b) Data distribution in *Iris* transformed by FWL



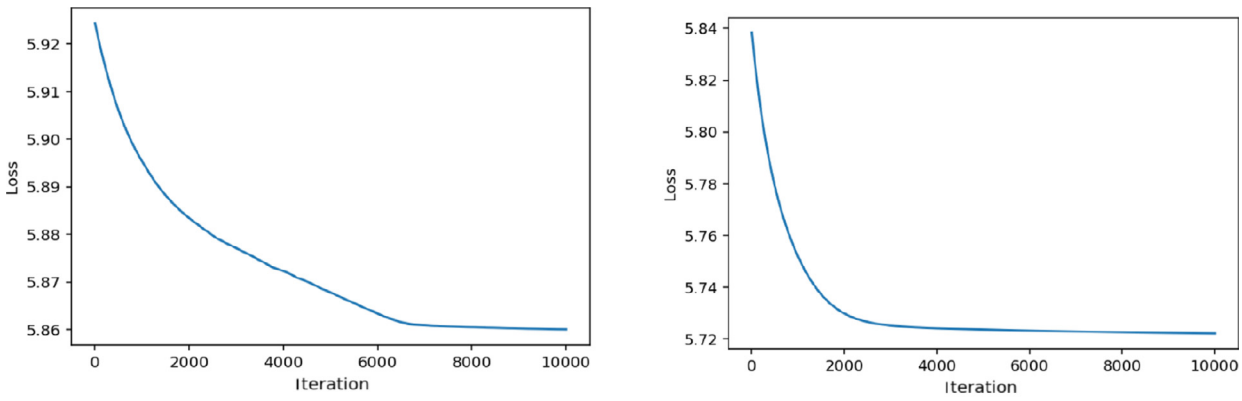
(c) Data distribution in *Iris* transformed by WML

Fig. 2. Data distribution in *Iris* before and after transformation.

3.3. Lower fuzziness of similarity matrix

WML can significantly reduce the fuzziness of the similarity matrix. Mathematically, minimizing the objective function

$$\min_{W\in\mathcal{R}^{n\times n}}\left[\rho_{pq}^{(W)}(1-\rho_{pq}^{(I)})+\rho_{pq}^{(I)}(1-\rho_{pq}^{(W)})\right] \tag{27}$$



(a) The loss of FWL (b) The loss of WML

Fig. 3. The loss of FWL and WML.

implies that

$$\begin{cases} \rho_{pq}^{(W)} \downarrow 0 & (\text{if } \rho_{pq}^{(I)} < 0.5) \\ \rho_{pq}^{(W)} \uparrow 1 & (\text{if } \rho_{pq}^{(I)} > 0.5) \end{cases}, \tag{28}$$

which makes

$$\text{Fuzziness}(\rho_{\mathbb{S}}^{(W)}) \triangleq \left[\rho_{pq}^{(W)} \log \rho_{pq}^{(W)} + (1-\rho_{pq}^{(W)}) \log(1-\rho_{pq}^{(W)})\right] \tag{29}$$

go to its minimum. Similar to the analysis for uncertainty, it is easy to validate that the decrease of fuzziness in WML is more significant than that in FWL.

3.4. Time complexity

WML requires fewer iterations than FWL. In WML, there are more elements in the transformation matrix and the range of values for the elements is enlarged, which allow the objective function to find a better solution in each iteration and speed up the training. Continue to consider the data set given in (23). After training by both FWL and WML, we get the trend graphs of the loss function as shown in Fig. 3, where the learning rate η used in both methods is 0.01. From this figure, we can see that the loss function of WML has converged after about 2300 iterations, but the loss function of FWL can only converge after 6500 iterations. Furthermore, the objective function in WML has a minimum generally smaller than that of FWL. However, it is noteworthy that although WML requires fewer iterations than FWL, it is computationally intensive and requires longer time in each iteration. As a solution, we can use parallel technique to reduce the running time of each iteration for WML. For example, in Fig. 1, $\vec{x}_p$ is taken as an input sample. The transformation for $\vec{x}_p$, i.e.,

$$\vec{y}_p = W_{n\times n}\vec{x}_p, \tag{30}$$

has a complexity of $O(n^2)$. This transformation operation can be well parallelized, i.e.,

$$\begin{cases} y_{1p} = \vec{w}_1^{\top}\vec{x}_p \\ y_{2p} = \vec{w}_2^{\top}\vec{x}_p \\ \vdots \\ y_{np} = \vec{w}_n^{\top}\vec{x}_p \end{cases}, \tag{31}$$

where $\vec{w}_i^{\top}$ is the ith row of $W_{n\times n}$, y_{ip} is the ith element of $\vec{y}_p$, and $i = 1, \ldots, n$. Thus, the transformation for a sample in parallel WML has a complexity of $O(n)$, which is the same as that in FWL. In the feedback process of neural network, such parallel method can also be used to reduce the running time of WML. That is, there is no obvious difference between the time complexity of WML under parallel mechanism and the time complexity of FWL in each iteration.

In summary, compared with FWL, the proposed WML has a stronger feature transformation capability, it can obtain a better similarity matrix with less uncertainty, and the time complexity in parallel WML is low.

4. Empirical studies

In order to verify the advantages of WML, in this section, we treat WML as a data pre-processing technique, and use the transformed data as the input for both unsupervised and supervised learning, i.e., clustering and classification.

4.1. Selected models for validation

For clustering, we know that k-means is one of the most classical algorithms. It determines the cluster indices of samples based on distances. Basically, samples close to each other will be assigned to the same cluster, and samples far away from each other will be assigned to different clusters. As analyzed in Section 2.2, the proposed WML can make nearly located samples closer and far-away samples farther away. From this perspective, it can improve the performance of k-means. Therefore, we choose k-means as a representative of clustering algorithms to verify the feasibility of WML.

For classification, there are many algorithms from different theoretical and practical perspectives. As analyzed in Section 3, the proposed WML can be represented as a FFNN. Thus, we choose FFNN for the experiment. Specifically, we adopt the most classical Backpropagation (BP) based algorithm, and a fast variant, i.e., random weight neural network (RWNN). Besides, we also want to verify that whether WML is effective when the supervised learning algorithm has nothing to do with sample distances or similarity metrics. As a representative of this category, C4.5 is used, which is a well-known decision tree model consists of a set of IF-THEN rules.

The details of the selected algorithms are described as follows.

- k-means: The main idea of k-means is to cluster with k points in space, which are called centroids. It calculates the Euclidean distance between the k centroids and each sample, and assign each sample to the cluster with the closest centroid [10]. Average of samples in each cluster is calculated as a new centroid iteratively until the best clustering result is obtained. The implementation of k-means is described in Algorithm 2.

Algorithm 2: k-mean clustering algorithm.

Input: Training set $\mathbb{S} = \{\vec{x}_i | \vec{x}_i \subset \mathcal{R}^n, i = 1, \cdots, N\}$, number of clusters k.
Output: Centroid vectors $\vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \ldots, \vec{\mu}^{(k)}$.

Randomly select k samples from $\mathbb{S}$ as the initial centroids $\vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \ldots, \vec{\mu}^{(k)}$;
repeat
  for *each* $\vec{x}_i$, $i = 1, \ldots, N$ **do**
    Compute $d^2(\vec{x}_i, \vec{\mu}^{(j)})$ where $j = 1, \ldots, k$;
    Mark the cluster for $\vec{x}_i$ by $\text{argmin}_{j=1,\ldots,k} d^2(\vec{x}_i, \vec{\mu}^{(j)})$;
  end
  for $j = 1, \ldots, k$ **do**
    Recalculate the centroid for each cluster $\mathcal{C}_j$, i.e., $\vec{\mu}^{(j)} = \frac{1}{|\mathcal{C}_j|} \sum_{\vec{x} \in \mathcal{C}_j} \vec{x}$;
  end
until $\vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \ldots, \vec{\mu}^{(k)}$ have no change;
return Centroid vectors $\vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \ldots, \vec{\mu}^{(k)}$.

- FFNN: FFNN is formed by a plurality of M-P neurons [15] connected in a hierarchical structure. It consists of one input layer, one or multiple hidden layers, and one output layer. BP is the most classical training algorithm, which adopts gradient-descent technique to adjust the connection weights and biases between neurons according to the training data. There are several parameters in FFNN, i.e., number of hidden layers, number of neurons in each hidden layer, learning rate, and the activation function. The implementation details of FFNN is omitted here due to its complexity.
- RWNN: RWNN was first proposed in [17,20] for single-hidden layer FFNN. It was improved in [16,25] as a non-iterative training algorithm in which the hidden weights and biases are randomly selected while the output weights are obtained analytically. The implementation of RWNN is described in Algorithm 3, and more details of RWNN can be referred to the literatures [3,8,9,11,14,22,31].

Algorithm 3: RWNN training algorithm.

Input: Training set $\mathbb{S} = \{(\vec{x}_i, \vec{t}_i)\}_{i=1}^N \subset \mathcal{R}^n \times \{0, 1\}^C$; activation function $g(x)$; number of hidden node $\tilde{N}$.
Output: Input weights w_j and biases b_j, and output weights α.

Randomly assign input weights w_j and biases b_j where $j = 1, 2, \ldots, \tilde{N}$;
Calculate the hidden layer output matrix H;
Calculate the output weights $\alpha = H^{\dagger}T$ where $H^{\dagger}$ is the Moore–Penrose generalized inverse of matrix H.

- C4.5: C4.5 is a classical decision tree induction algorithm. It adopts the concept of entropy to measure the degree of uncertainty for an attribute, and the attribute with the maximum information gain or gain rate is selected for expanding a node [32]. The implementation of C4.5 is described in Algorithm 4 .

Algorithm 4: C4.5 decision tree induction algorithm.

Input: Training set $\mathbb{S} = \{(\vec{x}_i, \vec{t}_i)\}_{i=1}^{N} \subset \mathcal{R}^n \times \{0,1\}^C$; attribute set $A = \{a_j\}_{j=1}^{n}$; threshold number $\hat{N}$ to stop splitting a node.

Output: The constructed decision tree.

```
Initialize Ω as an empty set;
Consider the set of all examples as the root-node, and add it to Ω;
while Ω is not empty do
    Select a node from Ω, denoted by 𝕏;
    if |𝕏| < N̂ then
        Treat 𝕏 as a leaf node and assign it label l* = argmax_{l=1,...,C} p_l;
    else
        Select expanding attribute a_* ∈ A with the maximum gain rate;
        for Each value a_*^v in a_* do
            Denote 𝕏_v as a subset of samples in 𝕊 with a_* = a_*^v;
            if all the examples in 𝕏_v come from the same class l* then
                Treat 𝕏_v as a leaf node and assign it label l*;
            else
                Add 𝕏_v to Ω;
            end
        end
    end
    Remove 𝕏 from Ω;
end
return The constructed decision tree.
```

4.2. Evaluation indexes

In general, classification results can be evaluated by the most commonly used metrics like training accuracy, testing accuracy, precision, recall, and F1-score, etc. While the evaluation on clustering results will be harder due to the unavailability of ground truth. In the following, we will introduce several evaluation indexes for clustering.

Clustering validation indexes can be classified into three categories: internal, external and relative. Rendón et al. [18] pointed out that the internal indexes can reflect the performance more accurately than external ones. Thus, we only focus on four internal indexes.

Davies–Bouldin Index (DBI). *DBI* [6] reflects the compactness of the data in the same cluster and the sparseness of the data in different clusters, which is defined as

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{j=1,\ldots,k, j\neq i} \frac{\frac{1}{|\mathcal{C}_i|}\sum_{\vec{x}\in\mathcal{C}_i} d(\vec{x}, \vec{\mu}^{(i)}) + \frac{1}{|\mathcal{C}_j|}\sum_{\vec{x}\in\mathcal{C}_j} d(\vec{x}, \vec{\mu}^{(j)})}{d(\vec{\mu}^{(i)}, \vec{\mu}^{(j)})}, \tag{32}$$

where k is the number of clusters, $\mathcal{C}_i$ and $\mathcal{C}_j$ represent the ith and the jth clusters, $\vec{\mu}^{(i)}$ and $\vec{\mu}^{(j)}$ represent the centroid vectors of $\mathcal{C}_i$ and $\mathcal{C}_j$, and $d(\cdot, \cdot)$ is a distance metric. A smaller value of *DBI* indicates a better clustering performance.

Dunn Index (DI). *DI* [7] reflects the compactness of the two closest clusters and the sparseness of the sparsest cluster, which is defined as

$$DI = \frac{\min_{\vec{x}\in\mathcal{C}_i, \vec{x}'\in\mathcal{C}_j, i\neq j} d(\vec{x}, \vec{x}')}{\max_{1\leq j\leq k} \max_{\vec{x}, \vec{x}'\in\mathcal{C}_j} d(\vec{x}, \vec{x}')}, \tag{33}$$

where k is the number of clusters, $\mathcal{C}_i$ and $\mathcal{C}_j$ represent the ith and the jth clusters, and $d(\cdot, \cdot)$ is a distance metric. A larger value of *DI* represents a better clustering result.

Calinski–Harabasz Index (CHI). *CHI* [2] reflects the quality of the clustering results by calculating the scatter matrix inside a cluster and the scatter matrix between different clusters. The *CHI* is defined as

$$CHI = \frac{\mathrm{trace}(S_B)}{\mathrm{trace}(S_W)} \cdot \frac{N-1}{N-k}, \tag{34}$$

where S_B and S_W are the between-cluster scatter matrix and the within cluster scatter matrix, N is the number of samples, and k is the number of clusters. We have $\mathrm{trace}(S_B) = \sum_{i=1}^{k} |\mathcal{C}_i| \times d^2(\vec{\mu}^{(i)}, \vec{\mu})$ and $\mathrm{trace}(S_W) = \sum_{i=1}^{k}\sum_{\vec{x}\in\mathcal{C}_i} d^2(\vec{x}, \vec{\mu}^{(i)})$, where $\mathcal{C}_i$ is the ith cluster, $\vec{\mu}^{(i)}$ is the centroid of $\mathcal{C}_i$, $\vec{\mu}$ is the centroid of the entire data set, and $d(\cdot, \cdot)$ is a distance metric. A larger value of *CHI* indicates a better clustering result.

Table 1
Detailed information of selected data sets.

No.	Data set	# Instances	# Attributes	# Classes	Class distribution	Attribute type
1	abalone	4177	8	3	1307/1342/1528	Categorical/Integer/Real
2	Autism Screening Adult	704	21	2	514/190	Integer
3	auto-mpg	392	8	3	79/244/68	Categorical/Real
4	breast-cancer	682	10	2	443/239	Integer
5	breast-cancer-P	194	33	2	147/47	Integer/Real
6	BHP looding attack on OBS	1060	22	4	494/150/115/300	Integer
7	cmc	1473	9	2	1364/109	Categorical/Integer
8	credit	666	15	2	299/367	Categorical/Integer/Real
9	data_banknote_authentication	1372	5	2	761/610	Real
10	glass	214	9	6	29/69/76/17/9/14	Real
11	ionosphere	351	32	2	224/127	Integer/Real
12	page-blocks	5473	10	5	115/4912/329/28/88	Integer/Real
13	waveform	5000	21	3	1666/1666/1668	Real
14	waveform-+noise	5000	40	3	1666/1666/1668	Real
15	Wireless Indoor Localization	2000	7	4	500/500/500/500	Real

Silhouette Index (SI). *SI* [19] is an internal clustering indicator, which is applicable to tasks whose actual category information is unknown. Suppose that sample $\vec{x}_i$ is categorized into cluster $\mathcal{C}_p$. Let $d_1 = \frac{1}{|\mathcal{C}_p|}\sum_{\vec{x}\in\mathcal{C}_p} d(\vec{x},\vec{x}_i)$ be the average distance between sample $\vec{x}_i$ and all the samples in $\mathcal{C}_p$. Cluster $\mathcal{C}_q$ is the closest cluster to $\mathcal{C}_p$, i.e., $\mathcal{C}_q = \text{argmin}_{\mathcal{C}_j\neq\mathcal{C}_p} d(\vec{\mu}^{(j)},\vec{\mu}^{(p)})$. Let $d_2 = \frac{1}{|\mathcal{C}_q|}\sum_{\vec{x}\in\mathcal{C}_q} d(\vec{x},\vec{x}_i)$ be the average distance between sample $\vec{x}_i$ and all the samples in cluster $\mathcal{C}_q$. Then, *SI* for $\vec{x}_i$ is defined as

$$SI(\vec{x}_i) = \frac{d_2 - d_1}{\max(d_1, d_2)}, \tag{35}$$

and the overall *SI* value is calculated as

$$SI = \frac{1}{N}\sum_{i=1}^{N} SI(\vec{x}_i), \tag{36}$$

where N is the number of samples. A larger value of *SI* indicates a better clustering performance.

4.3. Experimental setup

We select 15 data sets from UCI machine learning repositoty[1] to conduct the empirical studies. The details of the data sets are shown in Table 1. 10×5-fold cross-validation was conducted. That means, we randomly split the data set into 5 equal parts, each time we use 4 parts for training and the other part for testing. This process is repeated for 10 times and the average results are investigated. The experiments are implemented in Python and run on a computer with the Deepin operating system, an i3-6100 CPU, and 8 GB of RAM. The learning rate η in FWL and WML is set as 0.01, the initial value of W in WML is set as I. The number of clusters k in the k-means algorithm is set as the number of labels in the data set. The number of hidden layer nodes $\tilde{N}$ in RWNN is set as 20. The number of hidden layers, the number of hidden nodes in each hidden layer, and the learning rate of FFNN are set as 1, 50, and 0.01 respectively. Sigmoid activation function is used in both RWNN and FFNN. The threshold number $\hat{N}$ in C4.5 is set as 1. Since WML cannot process the data with different attribute ranges very well, we perform Min-Max normalization for each feature. Specifically, for each feature, the input values are normalized to [0,1] by $1-(x_{\max}-x)/(x_{\max}-x_{\min})$, where $x_{\max}$ and $x_{\min}$ are the maximum and minimum values among all the samples with regard to the feature, and x is the value to be normalized. In this case, WML can be applied to attributes with the same range.

We apply k-means clustering algorithm directly on the input features of the data sets without using the label information, and apply RWNN, FFNN and C4.5 on the entire data sets with label information. For clustering, the four indexes introduced in Section 4.2 are used to evaluate the results, where Mahalanobis distance is used as the distance metric $d(\cdot,\cdot)$. As for classification, training accuracy, testing accuracy, precision, recall, and F1-score are used as the evaluation criteria.

4.4. Performance analysis for clustering

Table 2 reports the results of k-means clustering algorithm trained on the original data, data transformed by FWL, and data transformed by WML regarding the four internal evaluation indexes. The lower the score of index *DBI*, the better the clustering result, and the higher the scores of indexes *DI*, *SI* and *CHI*, the better the clustering result. It can be seen that regarding these four internal indexes, k-means with the proposed WML performs the best, and k-means with FWL performs

[1] http://archive.ics.uci.edu/ml/.

Table 2
Clustering result.

Data set	BDIo	BDIv	BDIm	DIo	DIv	DIm	SIo	SIv	SIm	CHIo	CHIv	CHIm
1	0.632	**0.623**	0.632	**2.711**	2.576	2.669	0.492	0.479	**0.493**	8972.904	8183.404	**9010.698**
2	0.509	0.651	**0.497**	3.595	2.566	**3.770**	0.626	0.502	**0.636**	1973.938	1195.186	**2030.390**
3	0.910	1.043	**0.416**	1.376	1.214	**3.891**	0.416	0.382	**0.670**	971.499	875.311	**2260.749**
4	0.352	0.361	**0.351**	3.939	3.763	**4.031**	**0.773**	0.764	**0.773**	3847.265	3564.813	**3869.452**
5	1.024	1.052	**1.020**	1.895	1.844	**1.907**	0.374	0.351	**0.374**	135.193	128.361	**136.362**
6	1.119	1.389	**0.804**	1.271	0.947	**2.020**	0.374	0.343	**0.459**	1862.613	1577.130	**2750.340**
7	1.045	1.474	**0.956**	1.792	1.318	**1.907**	0.364	0.308	**0.411**	1172.700	424.323	**1445.951**
8	0.968	0.968	**0.779**	2.037	2.037	**2.316**	0.409	0.409	**0.538**	588.437	588.437	**996.411**
9	**0.712**	0.734	**0.712**	2.524	2.289	**2.593**	0.494	0.444	**0.498**	2131.796	1655.016	**2190.536**
10	2.916	2.086	**0.854**	0.142	0.202	**0.619**	0.161	0.219	**0.556**	256.805	435.778	**773.423**
11	0.880	0.884	**0.833**	1.574	1.558	**1.891**	0.453	0.451	**0.495**	315.978	310.652	**377.790**
12	1.113	1.694	**0.933**	0.924	0.638	**0.937**	0.323	0.185	**0.344**	3719.115	1275.166	**4863.434**
13	0.753	0.752	**0.547**	2.507	2.537	**3.503**	0.373	0.371	**0.553**	9241.689	9194.467	**16911.388**
14	0.744	0.743	**0.536**	2.600	2.595	**3.668**	0.370	0.371	**0.562**	9060.994	9074.919	**17456.016**
15	0.664	0.652	**0.508**	2.130	2.178	**3.630**	0.485	0.495	**0.604**	571.023	582.997	**845.076**

Note: DBIo is the value of *DBI* on the original data; *DBIv* is the value of *DBI* on the data transformed by FWL; *DBIm* is the value of *DBI* on the data transformed by WML, and so on for *DI*, *SI*, and *CHI*.

similarly or even worse than original *k*-means. It is worth noting that WML achieved much better results on data set 3, 6, 10, 12, 13, 14 and 15. By observing Table 1, a common feature of these data sets can be found, i.e., they have a relatively large number of clusters. Thus, a preliminary conclusion is that WML is good at processing data with a cluster number greater than or equal to 3. Figs. 4–7 further provide the visualizations of the clustering results regarding the four evaluation indexes. It is noticed that the advantage of *k*-means algorithm with WML becomes very obvious by the visualization, especially on the problems like data sets 3, 6, 10, 12, 13, 14 and 15.

Essentially, *k*-means clustering algorithm determines the cluster indices of samples based on distances. Samples close to each other will be assigned to the same cluster, and samples far away from each other will be assigned to different clusters. As analyzed in Section 2.2, WML can make nearly located points closer and far-away points farther away. Among the four indexes, *DBI*, *DI* and *SI* are distance-based metrics. Thus, WML can improve the performance of *k*-means regarding these indexes. Moreover, it is indicated in [2] that data with lower within-cluster distance and higher between-cluster distance can get a higher *CHI* score. Since WML can generally reduce the within-cluster distance and increase the between-cluster distance, the performance regarding *CHI* score can also be improved. Finally, it is concluded that WML can significantly improve the performance of *k*-means regarding all the considered indexes.

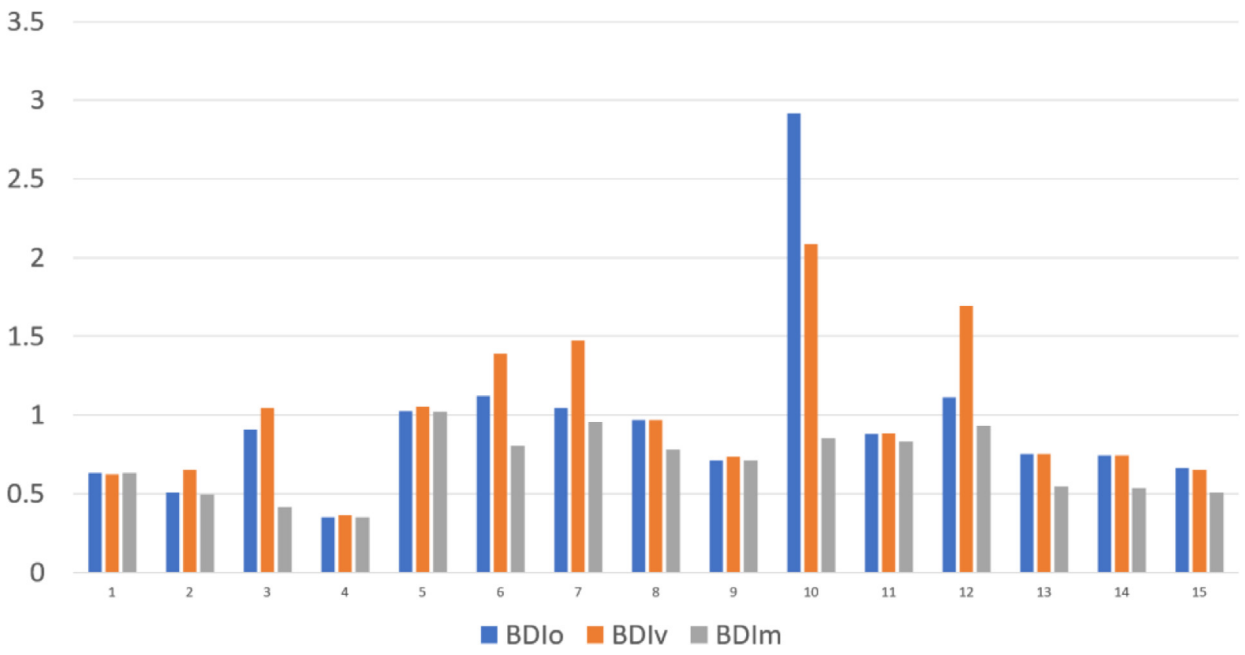


Fig. 4. Visualization of clustering results regarding *DBI* (the lower, the better).

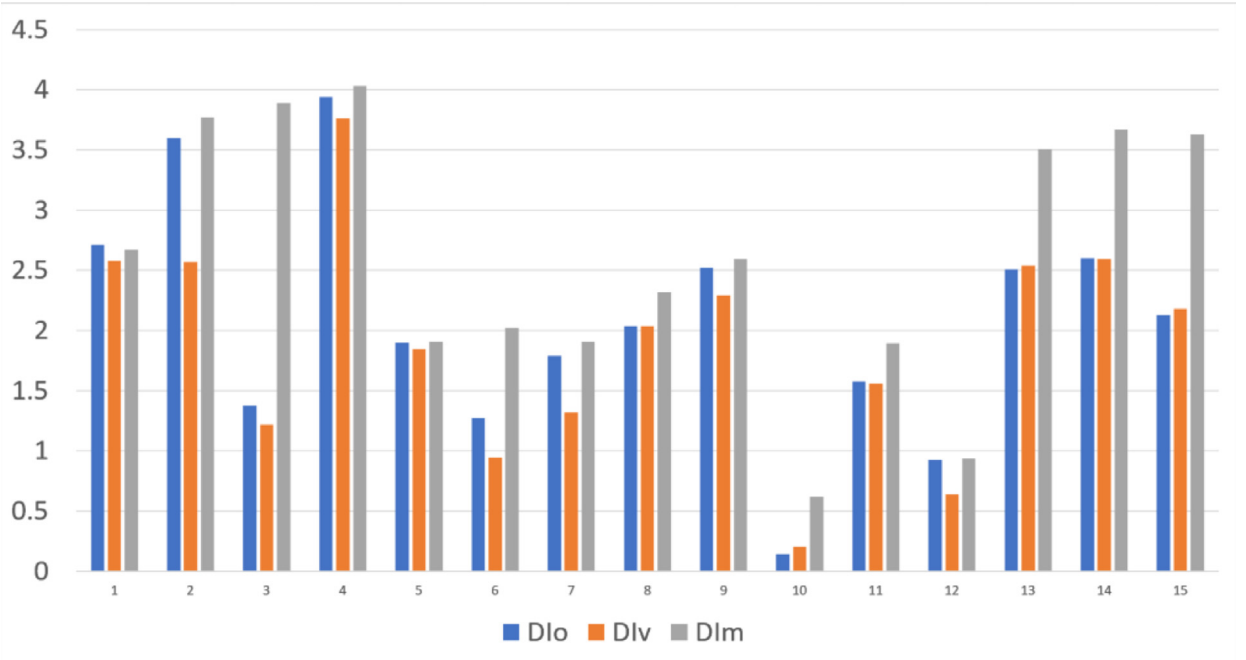


Fig. 5. Visualization of clustering results regarding *DI* (the higher, the better).

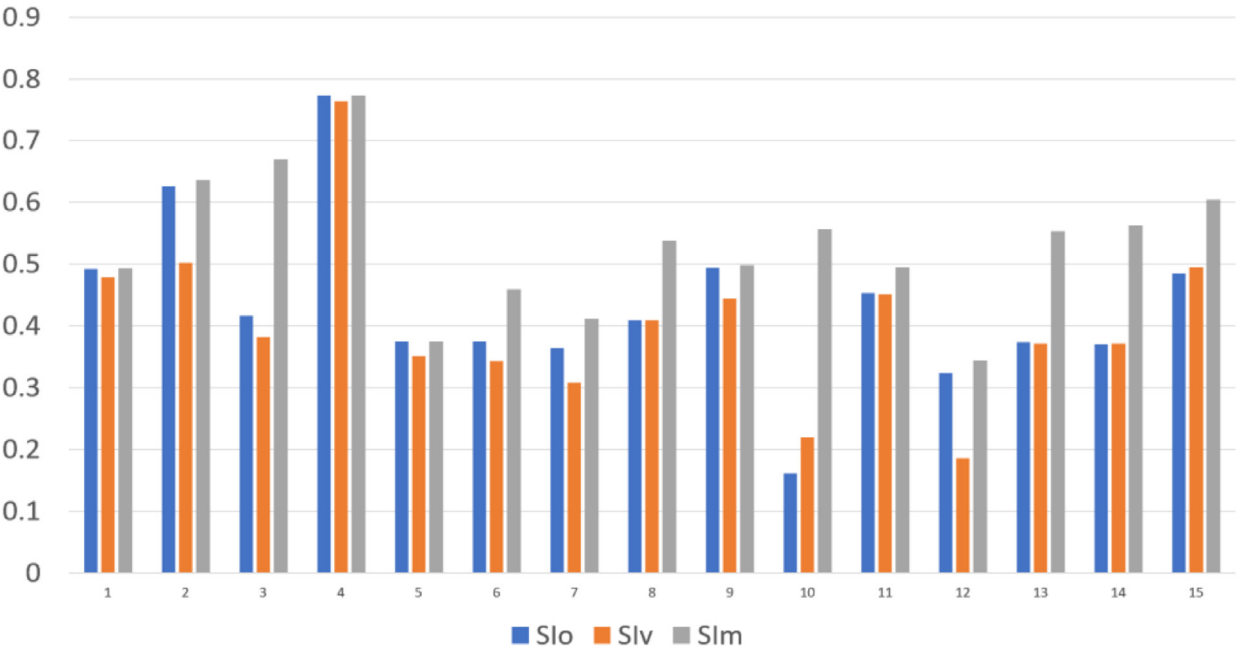


Fig. 6. Visualization of clustering results regarding *SI* (the higher, the better).

4.5. Performance analysis for classification

WML can generate a better similarity matrix with lower uncertainty for the data, and the data with lower uncertainty can help improve the performance of some supervised learning algorithms.

Tables 3 and 4 report the training and testing accuracy of RWNN, FFNN, and C4.5 on the original data, data transformed by FWL, and data transformed by WML, respectively. It is observed from Table 3 that the model learned on the original data obtains the best training accuracy in most cases. However, regarding the testing accuracy in Table 4, this advantage of the original data disappears. For RWNN, FFNN, and C4.5, the one using WML obtains the best result on half or more than half

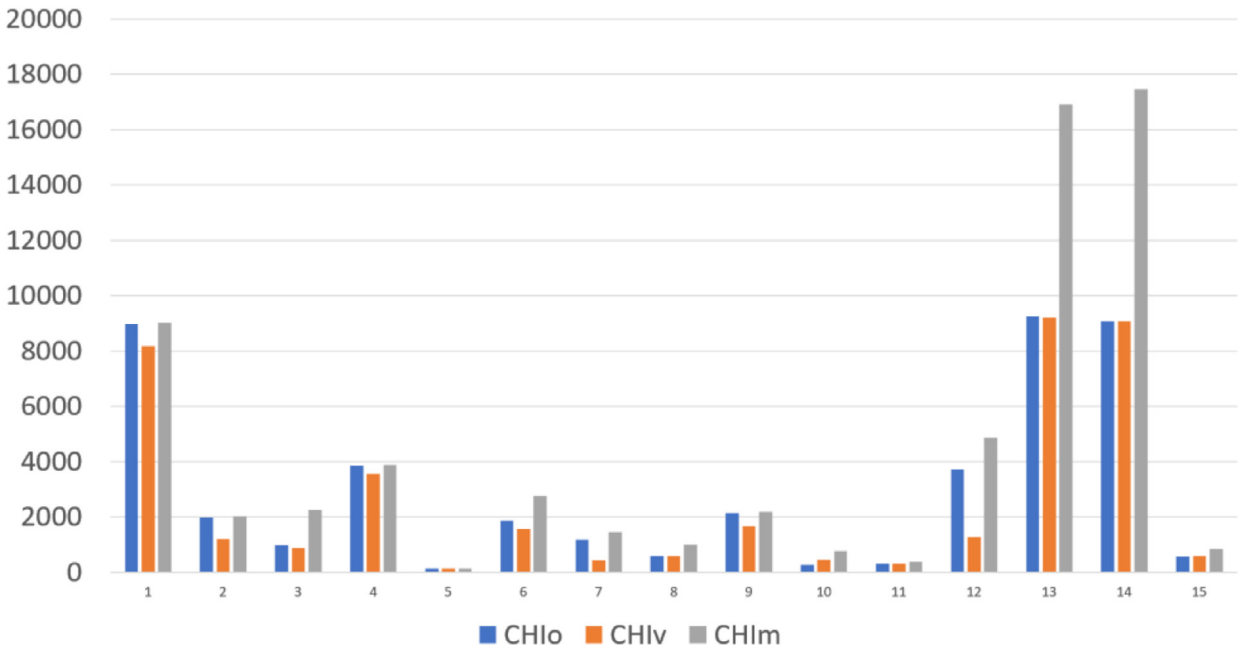


Fig. 7. Visualization of clustering results regarding *CHI* (the higher, the better).

Table 3
Classification result: training accuracy.

Data set	RWNNo	RWNNv	RWNNm	FFNNo	FFNNv	FFNNm	C4.5o	C4.5v	C4.5m
1	**0.582**	0.568	0.572	**0.541**	0.535	0.536	1.000	1.000	1.000
2	**0.958**	0.947	0.944	**0.984**	0.983	0.955	1.000	1.000	1.000
3	**0.854**	0.785	0.783	**0.736**	0.724	0.701	1.000	1.000	1.000
4	**0.978**	0.969	0.973	0.971	0.972	**0.973**	1.000	1.000	1.000
5	**0.831**	0.812	0.773	**0.847**	0.843	0.779	1.000	1.000	1.000
6	**0.903**	0.842	**0.903**	**0.721**	0.697	0.710	1.000	1.000	1.000
7	**0.926**	**0.926**	0.925	**0.926**	**0.926**	0.925	1.000	1.000	1.000
8	**0.882**	0.854	0.874	0.871	**0.874**	0.871	1.000	1.000	1.000
9	**0.999**	0.998	0.997	0.978	0.955	**0.981**	1.000	1.000	1.000
10	**0.798**	0.734	0.727	**0.664**	0.616	0.582	1.000	1.000	1.000
11	**0.854**	0.808	0.814	**0.925**	0.917	0.898	1.000	1.000	1.000
12	**0.956**	0.933	0.943	**0.913**	0.906	0.898	0.998	0.998	0.998
13	**0.852**	0.825	0.850	**0.855**	0.851	0.830	1.000	1.000	1.000
14	**0.818**	0.764	0.795	**0.846**	0.844	0.829	1.000	1.000	1.000
15	**0.998**	0.982	0.997	**0.991**	0.985	0.965	1.000	1.000	1.000

Note: The method with subscript o represents the result on the original data; the method with subscript v represents the result on the data transformed by FWL; the method with subscript m represents the result on the data transformed by WML.

of the data sets. This observation indicates that the data transformed by WML might be useful for handling the over-fitting problem, thus improving the performance of some supervised learning algorithms.

Tables 5–7 further report the precision, recall, and F1-score of RWNN, FFNN, and C4.5 on the original data, data transformed by FWL, and data transformed by WML. For these evaluation indexes, RWNN with WML can achieve the best result on about half of the data sets, but the improvement of FFNN and C4.5 with WML is not satisfactory in some cases.

RWNN and FFNN are network structures that reflect the mapping relationship between the feature space of the data and the category space. The complexity of the feature space directly affects the performance of RWNN and FFNN. The proposed WML performs a linear transformation on the data, which can reduce the uncertainty of the similarity matrix. Noting that the uncertainty of the similarity matrix may be closely related to the complexity of the feature space, thus, decreasing the uncertainty of the similarity matrix may reduce the complexity of the feature space. However, the underlying mechanism is still not clear, and theoretical support is also missing. Thus, network structures with WML are difficult to analyze in the current stage, leading to unsatisfactory performance in some cases as shown in the tables. As for C4.5, it is a rule-based learning algorithm that iteratively selects the expanding attribute with the highest information gain rate. During the whole process, the learning has nothing to do with similarity matrix or distance evaluation, thus, from theoretical perspective,

Table 4
Classification result: testing accuracy.

Data set	RWNNo	RWNNv	RWNNm	FFNNo	FFNNv	FFNNm	C4.5o	C4.5v	C4.5m
1	**0.562**	0.557	0.555	0.527	0.526	**0.532**	**0.498**	**0.498**	0.491
2	0.951	0.939	**0.959**	0.970	**0.978**	0.966	**1.000**	**1.000**	0.970
3	**0.776**	0.733	0.737	0.682	0.684	**0.692**	**0.792**	**0.792**	0.689
4	0.967	0.964	**0.969**	0.961	0.969	**0.973**	0.939	0.939	**0.955**
5	0.687	0.715	**0.792**	0.769	0.772	**0.818**	0.618	0.618	**0.649**
6	0.882	0.795	**0.890**	0.686	0.686	**0.698**	**1.000**	**1.000**	**1.000**
7	**0.929**	0.920	0.926	0.930	0.920	**0.931**	**0.882**	**0.882**	0.871
8	0.875	0.843	**0.883**	0.868	0.866	**0.887**	**0.830**	**0.830**	0.810
9	0.996	**0.997**	0.995	0.977	0.964	**0.978**	**0.981**	**0.981**	**0.981**
10	0.600	**0.656**	0.640	**0.619**	0.642	0.544	**0.637**	**0.637**	0.623
11	0.801	0.765	**0.812**	**0.876**	0.865	0.873	**0.894**	**0.894**	0.851
12	**0.950**	0.930	0.946	**0.917**	0.900	0.902	**0.960**	**0.960**	0.939
13	0.841	0.817	**0.851**	0.848	**0.849**	0.830	0.751	0.751	**0.794**
14	**0.820**	0.753	0.792	**0.847**	0.846	0.832	0.749	0.749	**0.817**
15	0.961	0.947	**0.981**	**0.983**	0.961	0.972	0.944	0.944	**0.950**

Note: The method with subscript o represents the result on the original data; the method with subscript v represents the result on the data transformed by FWL; the method with subscript m represents the result on the data transformed by WML.

Table 5
Classification result: testing precision.

Data set	RWNNo	RWNNv	RWNNm	FFNNo	FFNNv	FFNNm	C4.5o	C4.5v	C4.5m
1	**0.555**	0.549	0.547	0.517	**0.518**	0.516	**0.500**	**0.500**	0.494
2	0.926	**0.954**	0.951	0.966	**0.973**	0.950	**1.000**	**1.000**	0.961
3	**0.702**	0.656	0.624	0.574	**0.579**	0.558	**0.735**	**0.735**	0.588
4	0.965	0.962	**0.966**	0.959	0.964	**0.969**	0.936	0.936	**0.954**
5	0.561	**0.581**	0.454	0.771	**0.776**	0.513	**0.530**	**0.530**	0.504
6	0.868	0.743	**0.878**	0.705	**0.724**	0.719	**1.000**	**1.000**	**1.000**
7	**0.465**	0.460	0.463	**0.465**	0.460	0.463	**0.572**	**0.572**	0.547
8	0.876	0.842	**0.884**	0.869	0.864	**0.888**	**0.831**	**0.831**	0.810
9	0.995	**0.997**	0.995	0.975	0.962	**0.976**	**0.981**	**0.981**	0.980
10	0.545	0.534	**0.548**	**0.399**	0.375	0.346	0.577	0.577	**0.610**
11	**0.862**	0.798	0.836	0.883	0.875	**0.914**	**0.889**	**0.889**	0.835
12	0.84	0.861	**0.878**	**0.334**	0.326	0.214	**0.785**	**0.785**	0.673
13	0.842	0.817	**0.851**	0.849	**0.850**	0.836	0.750	0.750	**0.794**
14	**0.821**	0.753	0.793	**0.848**	0.847	0.838	0.749	0.749	**0.818**
15	0.959	0.951	**0.980**	**0.982**	0.956	0.969	**0.953**	**0.953**	**0.953**

Note: The method with subscript o represents the result on the original data; the method with subscript v represents the result on the data transformed by FWL; the method with subscript m represents the result on the data transformed by WML.

Table 6
Classification result: testing recall.

Data set	RWNNo	RWNNv	RWNNm	FFNNo	FFNNv	FFNNm	C4.5o	C4.5v	C4.5m
1	**0.564**	0.558	0.557	0.526	0.523	**0.530**	**0.502**	**0.502**	0.494
2	**0.959**	0.895	0.946	0.959	**0.971**	0.966	**1.000**	**1.000**	0.964
3	**0.696**	0.646	0.616	0.554	**0.564**	0.559	**0.716**	**0.716**	0.583
4	0.964	0.960	**0.965**	0.955	0.969	**0.972**	0.931	0.931	**0.947**
5	0.529	**0.544**	0.494	**0.624**	0.609	0.548	**0.524**	**0.524**	0.501
6	0.880	0.779	**0.886**	**0.689**	0.660	0.675	**1.000**	**1.000**	**1.000**
7	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.586**	**0.586**	0.556
8	0.879	0.844	**0.886**	0.873	0.871	**0.887**	**0.828**	**0.828**	0.807
9	0.996	**0.998**	0.996	0.979	0.966	**0.980**	**0.981**	**0.981**	**0.981**
10	0.538	0.517	**0.549**	**0.442**	0.437	0.388	**0.620**	**0.620**	0.589
11	**0.736**	0.707	0.715	**0.850**	0.836	0.814	**0.883**	**0.883**	0.829
12	**0.589**	0.458	0.573	**0.269**	0.231	0.210	**0.795**	**0.795**	0.638
13	0.841	0.817	**0.851**	0.847	**0.849**	0.831	0.750	0.750	**0.794**
14	**0.821**	0.753	0.793	**0.847**	0.846	0.833	0.749	0.749	**0.818**
15	0.970	0.948	**0.984**	**0.985**	0.968	0.976	0.942	0.942	**0.959**

Note: The method with subscript o represents the result on the original data; the method with subscript v represents the result on the data transformed by FWL; the method with subscript m represents the result on the data transformed by WML.

Table 7
Classification result: testing F1-score.

Data set	RWNNo	RWNNv	RWNNm	FFNNo	FFNNv	FFNNm	C4.5o	C4.5v	C4.5m
1	**0.547**	0.540	0.545	**0.496**	0.488	0.494	**0.497**	**0.497**	0.491
2	0.952	0.937	**0.959**	0.970	**0.978**	0.966	**1.000**	**1.000**	0.970
3	**0.777**	0.732	0.727	0.669	**0.676**	0.672	**0.790**	**0.790**	0.688
4	0.967	0.964	**0.969**	0.960	0.969	**0.973**	0.938	0.938	**0.955**
5	0.640	0.662	**0.738**	0.724	0.726	**0.768**	0.616	0.616	**0.677**
6	0.875	0.768	**0.884**	0.664	0.670	**0.679**	**1.000**	**1.000**	**1.000**
7	**0.896**	0.881	0.891	**0.896**	0.881	0.891	0.885	0.885	**0.874**
8	0.875	0.844	**0.883**	0.868	0.866	**0.887**	**0.829**	**0.829**	0.809
9	0.996	**0.997**	0.995	0.977	0.964	**0.978**	**0.981**	**0.981**	**0.981**
10	0.594	**0.633**	0.606	0.564	**0.593**	0.479	**0.646**	**0.646**	0.618
11	**0.779**	0.741	**0.779**	**0.873**	0.861	0.864	0.894	0.894	**0.850**
12	**0.944**	0.915	0.936	**0.889**	0.863	0.857	**0.960**	**0.960**	0.939
13	0.841	0.816	**0.850**	0.847	**0.848**	0.828	0.751	0.751	**0.794**
14	**0.820**	0.752	0.790	**0.846**	0.845	0.830	0.749	0.749	**0.818**
15	0.961	0.947	**0.981**	**0.983**	0.962	0.972	0.944	0.944	**0.950**

Note: The method with subscript o represents the result on the original data; the method with subscript v represents the result on the data transformed by FWL; the method with subscript m represents the result on the data transformed by WML.

Table 8
Execution time of FWL and WML (Seconds).

Data set	FWL	WML	WML with 4 Threads
1	157.40	207.02	73.94
2	11.25	6.83	5.69
3	10.04	5.06	4.29
4	9.47	7.17	5.87
5	8.81	6.24	5.20
6	12.33	8.67	7.05
7	25.67	37.04	20.58
8	12.88	8.77	7.31
9	21.94	33.38	19.64
10	12.00	7.01	5.84
11	13.97	8.76	6.74
12	290.11	400.94	125.29
13	239.81	331.89	107.06
14	259.90	332.70	107.32
15	10.83	4.35	3.35

there is no supporting points for WML to improve the performance of C4.5, which is consistent with our experimental results.

Through the above analysis, it can be concluded that WML can significantly improve the performance of clustering algorithms like *k*-means. It can enhance the performance of classification algorithms like RWNN in some cases, but further researches are needed to discover the underlying reasons and theoretical supports.

4.6. Execution time

Finally, Table 8 reports the execution time of FWL, WML in serial, and WML in parallel on the data sets. In serial, WML has an advantage in running time on small data sets, but it is more time-consuming on larger ones. We use multi-threading technique to accelerate the training of WML. In parallel, the running time of WML is greatly reduced, which is much shorter than that of FWL.

5. Concluding remarks

The proposed WML method improves the existing FWL method by extending the weight vector into a square matrix and changes the calculation formula for similarity matrix. Such changes can enlarge the searchable range of the loss function, obtain a better similarity matrix, and enhance the data transformation capability. Experimental results show that WML performs better than FWL for some clustering algorithms like *k*-means, and improves the performance of some classification algorithms such as RWNN. There are few disadvantages of WML, i.e., (1) more variables need to be trained and the time complexity is higher under serial condition; (2) it needs to calculate the distance matrix of the data set, which is a memory consuming operation. Our future works regarding this topic are listed as follows: (1) the transformation matrix W is a square matrix, which means that the transformed data set has the same number of features as the original data set, discussing the

transformation with a non-square matrix will also be useful; (2) WML is essentially a linear transformation of the data, we will try to extend it to nonlinear case; (3) it will be interesting to discuss that which supervised learning algorithms are suitable for WML, and also try to discover the underlying reasons.

Conflict of interest

The corresponding author certifies that all authors have read the manuscript before submission, and there is no conflict of interest associated with this publication, and there has been no financial support for this work that could have influenced its outcome.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (Grant 61772344, Grant 61811530324, and Grant 61732011), in part by the HD Video R&D Platform for Intelligent Analysis and Processing in Guangdong Engineering Technology Research Centre of Colleges and Universities (Grant GCZX-A1409), in part by the Natural Science Foundation of Shenzhen (Grant JCYJ20170818091621856), in part by the Natural Science Foundation of SZU (Grant 827-000140, Grant 827-000230, and Grant 2017060), and in part by the Interdisciplinary Innovation Team of Shenzhen University.

References

[1] J. Basak, R.K. De, S.K. Pal, Unsupervised feature selection using a neuro-fuzzy approach, Pattern Recognit. Lett. 19 (11) (1998) 997–1006.
[2] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat.-Theory Methods 3 (1) (1974) 1–27.
[3] W. Cao, X. Wang, Z. Ming, J. Gao, A review on neural networks with random weights, Neurocomputing 275 (2018) 278–287.
[4] R.G. Cinbis, J. Verbeek, C. Schmid, Unsupervised metric learning for face identification in tv video, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 1559–1566.
[5] A.K. Das, Weighted fuzzy soft multiset and decision-making, Int. J. Mach. Learn.Cybern. 9 (5) (2018) 787–794.
[6] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach.Intell. PAMI-1 (2) (1979) 224–227.
[7] J.C. Dunn, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, J. Cybern. 3 (3) (1973) 32–57.
[8] L. Hu, Y. Chen, J. Wang, C. Hu, X. Jiang, Okrelm: online kernelized and regularized extreme learning machine for wearable-based activity recognition, Int. J. Mach. Learn.Cybern. 9 (9) (2018) 1577–1590.
[9] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[10] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient *k*-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach.Intell. 24 (7) (2002) 881–892.
[11] J. Liu, M.J.A. Patwary, X. Sun, K. Tao, An experimental study on symbolic extreme learning machine, Int. J. Mach. Learn.Cybern. (2018) 1–11.
[12] P. Liu, X. Liu, The neutrosophic number generalized weighted power averaging operator and its application in multiple attribute group decision making, Int. J. Mach. Learn.Cybern. 9 (2) (2018) 347–358.
[13] P. Liu, F. Teng, Multiple attribute decision making method based on normal neutrosophic generalized weighted power averaging operator, Int. J. Mach. Learn.Cybern. 9 (2) (2018) 281–293.
[14] D. Lowe, Multi-variable functional interpolation and adaptive networks, Complex Syst. 2 (1988) 321–355.
[15] P.W. McCulloch W S, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5(4) (1943) 115–133.
[16] Y.-H. Pao, G.-H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[17] Y.-H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, Computer 25 (5) (1992) 76–79.
[18] E. Rendón, I. Abundez, A. Arizmendi, E.M. Quiroz, Internal versus external cluster validation indexes, Int. J. Comput.Commun. 5 (1) (2011) 27–34.
[19] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[20] W.F. Schmidt, M.A. Kraaijveld, R.P.W. Duin, Feedforward neural networks with random weights, in: 11th IAPR International Conference on Pattern Recognition. Vol. II. Conference B: Pattern Recognition Methodology and Systems, IEEE, 1992, pp. 1–4.
[21] B. Wang, J. Jiang, W. Wang, Z.-H. Zhou, Z. Tu, Unsupervised metric fusion by cross diffusion, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 2997–3004.
[22] J. Wang, L. Zhang, J.-J. Cao, D. Han, Nbwelm: naive bayesian based weighted extreme learning machine, Int. J. Mach. Learn.Cybern. 9 (1) (2018) 21–35.
[23] X. Wang, G. Hua, T.X. Han, Discriminative tracking by metric learning, in: European Conference on Computer Vision, Springer, 2010, pp. 200–214.
[24] X. Wang, Y. Wang, L. Wang, Improving fuzzy c-means clustering based on feature-weight learning, Pattern Recognit. Lett. 25 (10) (2004) 1123–1132.
[25] X.-Z. Wang, T. Zhang, R. Wang, Noniterative deep learning: incorporating restricted boltzmann machine into multilayer random weight neural networks, IEEE Trans. Syst. Man Cybern. in press, DOI: 10.1109/TSMC.2017.2701419 (2018).
[26] X. Xu, S. Ding, M. Du, Y. Xue, Dpcg: an efficient density peaks clustering algorithm based on grid, Int. J. Mach. Learn.Cybern. (2018) 1–12.
[27] L. Yang, R. Jin, Distance metric learning: a comprehensive survey, Michigan State Univ. 2 (2) (2006) 4.
[28] J. Ye, Z. Zhao, H. Liu, Adaptive distance metric learning for clustering, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–7.
[29] J. Ye, Z. Zhao, H. Liu, Adaptive distance metric learning for clustering, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–7.
[30] D.S. Yeung, X. Wang, Improving performance of similarity-based clustering by feature weight learning, IEEE Trans. Pattern Anal. Mach.Intell. 24 (4) (2002) 556–561.
[31] H. Zhao, X. Guo, M. Wang, T. Li, C. Pang, D. Georgakopoulos, Analyze eeg signals with extreme learning machine based on pmis feature selection, Int. J. Mach. Learn.Cybern. 9 (2) (2018) 243–249.
[32] Z.-H. Zhou, Machine Learning, Beijing: Tsinghua University Press, 2016.