Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05343-2

ORIGINAL ARTICLE

A precise and stable machine learning algorithm: eigenvalue classification (EigenClass)

Uğur Erkan[1]

Received: 25 October 2019 / Accepted: 3 September 2020

Abstract
In this study, a precise and efficient eigenvalue-based machine learning algorithm, particularly denoted as Eigenvalue Classification (EigenClass) algorithm, has been presented to deal with classification problems. The EigenClass algorithm is constructed by exploiting an eigenvalue-based proximity evaluation. To appreciate the classification performance of EigenClass, it is compared with the well-known algorithms, such as k-nearest neighbours, fuzzy k-nearest neighbours, random forest (RF) and multi-support vector machines. Number of 20 different datasets with various attributes and classes are used for the comparison. Every algorithm is trained and tested for 30 runs through 5-fold cross-validation. The results are then compared among each other in terms of the most used measures, such as accuracy, precision, recall, micro-F-measure, and macro-F-measure. It is demonstrated that EigenClass exhibits the best classification performance for 15 datasets in terms of every metric and, in a pairwise comparison, outperforms the other algorithms for at least 16 datasets in consideration of each metric. Moreover, the algorithms are also compared through statistical analysis and computational complexity. Therefore, the achieved results show that EigenClass is a precise and stable algorithm as well as the most successful algorithm considering the overall classification performances.

Keywords Data classification · Eigenvalues · Learning algorithm · Machine learning · Supervised learning

1 Introduction

The field of data classification is of a growing importance due to the unpredictability, large amount, and complexity of real-world data which include multi-class predictions in practical applications [1–3]. The evolution of a new classification algorithm is an essential and challenging research topic in the field of machine learning [4–6]. Classification methods aim to predict a class label of input samples include a set of attributes [7]. Classification methods determine class labels of observed input test data according to training data. In classifying data, various mathematical distance calculations and intuitive methods are employed, and expert opinions are considered [8]. Classification problems with multiple classes and nonlinear class constraints with considerable numbers of training data which lead to computational cost generally require complex classifiers [9]. In classification methods, a class is assigned to an observed input test data by performing a learning process with training data. The learning process is divided into two categories, i.e. supervised and unsupervised learning [10]. The supervised classification consists of two stages. In the first stage, specific attributes are extracted from training data according to class labels, which aim to train a classifier model to be able to assign a class label to test data. The second stage is the prediction phase, classifying test data through a trained model [9, 11, 12].

Classification algorithms can be defined as two types based on the number of class labels: binary and multi-class classification. In recent years, the multi-class classification has been attracting more attention in engineering problems [13]. Bayesian [14], artificial neural networks (ANN), support vector machines (SVM) [15], k-Nearest Neighbours (kNN) [16], random forests (RF) [17] algorithms are commonly used as supervised learning techniques [18, 19]. SVM and k-NN are particularly specified as

✉ Uğur Erkan
ugurerkan@kmu.edu.tr

[1] Department of Computer Engineering, Engineering Faculty, Karamanoğlu Mehmetbey University, 70200 Karaman, Turkey

Published online: 17 September 2020

neighbourhood-based algorithms. In general, the term "nearest" is defined along with the concept of "distance" among samples in data space [20]. The nearest neighbour classifier chooses the observed datum according to the class label of the nearest neighbour. kNN is a well-known multi-class method applicable to data classification [21].

The selection of distance functions is key to kNN. Using different functions, such as Euclidean, Mahalanobis, Hamming and Minkowski, provides flexibility and versatility, thus yields more precise results [11]. SVM based on a statistical approach is another popular algorithm for binary classification and regression in a great many areas, such as image classification, text categorisation, and bioinformatics [22, 23]. SVM classifies data samples by constructing a hyperplane in an N-dimensional data space, in such a way that it attempts to maximise the margin between the data samples of class labels. The RF method, constituting many individual tree blocks that compose a random forest, classifies data samples through decision trees [17]. Each tree in a random forest produces a class label prediction, and then class with the highest number of votes would be the ultimate prediction result. Some individual trees might produce inaccurate predictions. The ultimate prediction is likely to be correct, thanks to the employed multiple trees. The review of the algorithms mentioned above manifests that the existing classification algorithms have their benefits and weaknesses depending on the intended applications. Nevertheless, more successful methods built on a new efficient basis would prove promising due to the need for the classification of increasingly diverse data sets.

Eigenvalues, also known as characteristic roots, is a special value of a linear system, associated with an eigenvector. It is widely utilised in such common phenomena as machine learning, physics, mathematics, and engineering. Eigenvalue, able to grasp a key factor, offers information about the degree of correlation between two symmetric matrices. Therefore, it has been successfully exploited in some size reduction processes, such as principal component analysis (PCA) [24] and linear discriminant analysis (LDA) [25], to increase calculation efficiency and prevent overfitting in high-dimensional data. The eigenvalue is a scale of a matrix which is attached to eigenvector providing an axis magnitude. Thus, PCA and LDA evaluate the covariance of data. Principal components are obtained in order of significance by ranking eigenvalues from the highest to the lowest. A summary matrix can thus be achieved by multiplying the data with the determined eigenvector. Although the concept of eigenvalue has the outstanding efficiency of exploring the correlation in data analysis, it has not been yet exploited in terms of classification algorithms. The application of the concept of eigenvalue to a classification algorithm, which is a novel approach, can be an encouraging practice.

In this study, a machine learning algorithm exploiting the generalised eigenvalue concept, namely eigenvalue classification (EigenClass) algorithm, has been developed for both binary and multi-class problems by referring to the proximity between the test and the training data. The classification label of the observed test data is determined by calculating the eigenvalues of each sample in the test data with respect to the training data. To appreciate the classification performance of EigenClass, it is compared with the well-known machine learning algorithms kNN, Fuzzy kNN [26], RF, and Multi SVM (MSVM). The results are then compared among each other in terms of the most used measures, such as accuracy, precision, recall, micro-F-measure, and macro-F-measure. The proposed EigenClass classifies datasets with higher values in the occurrence of 15 datasets more competently than the other algorithms in view of all the measures. The results demonstrate that the proposed EigenClass is an outstanding and useful algorithm for the classification problems.

In the present study, in Sect. 2, the preliminaries and algorithm steps of the proposed EigenClass are presented, and some basic notions are supplied. In Sect. 3, a comparison appreciating the performance of the EigenClass is performed by comparing it with well-known algorithms. Finally, we discuss EigenClass and the need for further research.

2 Preliminaries and EigenClass algorithm

The generalised eigenvalue is a handy and versatile mathematical tool that provides information about the correlation of linear transformations. In this section, firstly, some basic notions related to the EigenClass algorithm are presented. Throughout this paper, $A = [a_{ij}]$ denotes a data matrix which has order $m \times n$, where m and n stand for the number of the attributes and the number of the samples in the data matrix, respectively. A_{train} represents the training matrix obtained from A, where the last column contains class labels of the data. A_{test} indicates the test matrix obtained from A. A^r symbolises the extracted matrix for r-class of A, where $r = 1, 2, \ldots, l$ is the number of class. $A^r_{i-\text{train}}$ and $A_{i-\text{test}}$ refer to ith row of A^r_{train} and A_{test}, respectively.

The following definition describes the concept of generalised eigenvalue utilised for an evolving the EigenClass algorithm.

Definition 2.1 Let A and B be two matrices, and $\boldsymbol{x}$ be a nonzero n-dimensional vector. If there exists a scalar λ such that $A\boldsymbol{x} = \lambda B\boldsymbol{x}$, then λ is called generalised eigenvalue of A according to B or briefly eigenvalue of A according to B. The vector which contains all eigenvalues of A according to B is denoted by $\text{eig}(A, B)$.

It must be noted that if A and B be two diagonal matrices whose diagonal entries differ from zero, then the generalised eigenvalues λ therein are real numbers and $\mathrm{eig}(A,A) = [1\ldots1]^T$. Throughout this paper, for the matrix $A = [a_{ij}]$, $\sum A = \sum_{i,j} a_{ij}$ and $|A| = [|a_{ij}|]$, where $|.|$ is absolute value function. In other words, $\sum A$ means the sum of all the entries of A and that $|A|$ denotes a matrix whose entries are equal to the absolute values of the entries of A.

Definition 2.2 Let A and B be two diagonal matrices whose diagonal entries differ from zero. Then,

$$q(A,B) = \sum \left| \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} - \mathrm{eig}(A,B) \right|$$

is called A's quasi-distance to B.

We can consider that A will be close B when $q(A,B)$ is close to zero. For example, if $A = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$, and $C = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, then, $\mathrm{eig}(A,A) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mathrm{eig}(A,B) = \begin{bmatrix} 1 \\ 1.25 \end{bmatrix}$, $\mathrm{eig}(A,C) = \begin{bmatrix} 1 \\ 2.5 \end{bmatrix}$, $q(A,B) = 0.25$, and $q(A,C) = 1.5$. Therefore, in the present paper, we accept that A is closer to B than C.

To determine the eigenvalues of a matrix, the matrix should be in a square form. However, each sample in a dataset given for classification is generally represented as a row of a data matrix. To overcome this difficulty, we convert each sample of the data matrix into a diagonal matrix.

Definition 2.3 Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. Then, the diagonal form of $\boldsymbol{x}$, namely $\mathrm{diag}(\boldsymbol{x})$, as follows:

$$\begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix}$$

The following pseudocode describes the working principle of the EigenClass algorithm step by step. Once the training matrix A_{train} and test matrix A_{test} are read (Step 1), a very small value (0.0001) is assigned instead of zeros (Step 2). The reason for this assignment is the need for nonzero elements for the calculation of the generalised eigenvalues. For all r, the r-class matrices A^r_{train} are extracted from the training matrix A_{train} (Step 3). For all i, r and t, the generalised eigenvalues for $\mathrm{diag}(A^r_{t-\text{train}})$ and $\mathrm{diag}(A_{i-\text{test}})$ are calculated and then the matrix B^r is constructed by $\mathrm{diag}(A^r_{t-\text{train}})$'s quasi-distance values to $\mathrm{diag}(A_{i-\text{test}})$ (Step 4). The elements in each row of B^r are rearranged in an ascending rank (Step 5). Recall that minimum values mean maximum correlation. A column matrix is formed via k-mean of each ascended row to improve the degree of proximity (Step 6). In other words, k is the degree of freedom by which the algorithm can be tuned. Finally, the classification label of the samples in the test data is found by operationalising the row number corresponding to the minimum element in the column matrix (Step 7). This way can determine the class labels of all the samples in the test data.

The pseudocode of the EigenClass algorithm

Step 1. Read a nonempty train matrix A_{train} and test matrix A_{test} of A

Step 2. For all i and j,

If $a_{ij-train} = 0$

$a_{ij-train} \leftarrow 0.0001$

If $a_{ij-test} = 0$

$a_{ij-test} \leftarrow 0.0001$

Step 3. For all r

Obtain training matrices with r-class A^r_{train}

Step 4. For all i

Obtain the matrix $B^i = [b^i_{rt}]$ for i^{th} row of A_{test} such that

$$b^i_{rt} = q\left(\mathrm{diag}(A^r_{t-train}), \mathrm{diag}(A_{i-test})\right)$$

Step 5. For all i

Obtain ascending-sorted matrix $\overline{B}^i$ of B^i to calculate its k-mean

Step 6. For all i

Obtain $C^i = [c^i_{s1}]_{r\times 1}$ such that $c^i_{s1} = k$-mean of $\overline{B}^i$

Step 7. For all i

Obtain the matrix $D = [d_{1i}]$, which denotes the assigned new classes to A_{test}. Here, $d_{1i} = \mathrm{argmin}_s\, c^i_{s1}$

To further increase the intelligibility of the EigenClass algorithm regarding the pseudocode, a numerical example is supplied below,

Example 2.1 A data matrix A taken from Balance Scale dataset [27] is provided below to implement EigenClass. It has 15 samples with 3 classes ($l = 3$) which are in the last column, and each class has 5 samples.

$$A = \begin{bmatrix} 1 & 3 & 1 & 3 & 1 \\ 2 & 5 & 2 & 5 & 1 \\ 2 & 4 & 2 & 4 & 1 \\ 2 & 1 & 1 & 2 & 1 \\ 3 & 1 & 1 & 3 & 1 \\ 1 & 1 & 3 & 2 & 2 \\ 2 & 1 & 2 & 3 & 2 \\ 2 & 3 & 2 & 4 & 2 \\ 3 & 3 & 2 & 5 & 2 \\ 3 & 1 & 3 & 2 & 2 \\ 1 & 4 & 1 & 3 & 3 \\ 1 & 5 & 1 & 1 & 3 \\ 2 & 2 & 1 & 1 & 2 \\ 3 & 3 & 1 & 3 & 3 \\ 3 & 3 & 1 & 4 & 3 \end{bmatrix}$$

Step 1

$$A_{\text{train}} = \begin{bmatrix} 1 & 3 & 1 & 3 & 1 \\ 2 & 4 & 2 & 4 & 1 \\ 2 & 1 & 1 & 2 & 1 \\ 3 & 1 & 1 & 3 & 1 \\ 1 & 1 & 3 & 2 & 2 \\ 2 & 1 & 2 & 3 & 2 \\ 2 & 3 & 2 & 4 & 2 \\ 3 & 1 & 3 & 2 & 2 \\ 1 & 4 & 1 & 3 & 3 \\ 1 & 5 & 1 & 1 & 3 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 3 & 1 & 3 & 3 \end{bmatrix}$$

and

$$A_{\text{test}} = \begin{bmatrix} 2 & 5 & 2 & 5 \\ 3 & 3 & 2 & 5 \\ 3 & 3 & 1 & 4 \end{bmatrix}$$

Step 2 Since no element with zero value exists, there is no need to apply the procedure in this step.

Step 3 The r-class training matrices belonging to 1-class, 2-class, and 3-class are extracted from the training matrix A_{train} as given below.

$$A^1_{\text{train}} = \begin{bmatrix} 1 & 3 & 1 & 3 \\ 2 & 4 & 2 & 4 \\ 2 & 1 & 1 & 2 \\ 3 & 1 & 1 & 3 \end{bmatrix},$$

$$A^2_{\text{train}} = \begin{bmatrix} 1 & 1 & 3 & 2 \\ 2 & 1 & 2 & 3 \\ 2 & 3 & 2 & 4 \\ 3 & 1 & 3 & 2 \end{bmatrix},$$

$$A^3_{\text{train}} = \begin{bmatrix} 1 & 4 & 1 & 3 \\ 1 & 5 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 3 & 3 & 1 & 3 \end{bmatrix}$$

Step 4 For $A^1_{1-\text{train}} = [1 \quad 3 \quad 1 \quad 3]$ and $A_{1-\text{test}} = [2 \quad 5 \quad 2 \quad 5]$, the generalised eigenvalues of $\text{diag}\left(A^1_{1-\text{train}}\right)$ and $\text{diag}\ (A_{1-\text{test}})$ are calculated as follows:

$$\begin{aligned} &\text{eig}\left(\text{diag}\left(A^1_{1-\text{train}}\right), \text{diag}\ (A_{1-\text{test}})\right) \\ &= \text{eig}\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}\right) = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.6 \\ 0.6 \end{bmatrix} \end{aligned}$$

Then, b^1_{11} of the proximity matrix B^1 is calculated as follows:

$$\begin{aligned} b^1_{11} &= q\left(\text{diag}\left(A^1_{1-\text{train}}\right), \text{diag}(A_{1-\text{test}})\right) \\ &= 0.5000 + 0.5000 + 0.4000 + 0.4000 = 1.8000 \end{aligned}$$

The other entries of B^1 are obtained similarly. Thus,

$$B^1 = \begin{bmatrix} 1.8 & 0.4 & 1.9 & 2.2 \\ 2.4 & 1.2 & 0.6 & 2.4 \\ 1.6 & 1.8 & 1.9 & 1.8 \end{bmatrix}$$

Step 5 The following matrix $\bar{B}^1$ is achieved by sorting each row of B^1 in an ascending rank to compute k-means of the closest k training data to test data in each row of B^1.

$$\bar{B}^1 = \begin{bmatrix} 0.4 & 1.8 & 1.9 & 2.2 \\ 0.6 & 1.2 & 2.4 & 2.4 \\ 1.6 & 1.8 & 1.8 & 1.9 \end{bmatrix}$$

Step 6 In the case of $k = 3$, we obtain $C^1 = \begin{bmatrix} 1.3667 \\ 1.4000 \\ 1.7333 \end{bmatrix}$ via 3-mean of $\bar{B}^1$ such that $C^1_{11} = \frac{0.4+1.8+1.9}{3} = 1.3667$, $C^1_{12} = \frac{0.6+1.2+2.4}{3} = 1.4000$, and $C^1_{13} = \frac{1.6+1.8+1.8}{3} = 1.7333$.

Step 7 Since the minimum value is 1.3667 in C^1, $d_{1i} = 1$ that is $A_{1-\text{test}}$ should belong to the 1-class.

Table 1 The properties of 20 datasets from UCI

No	Name	Sample	Attribute	Class
1	Seeds	210	7	3
2	Wireless	2000	7	4
3	Wine	178	13	3
4	Immunotherapy	90	7	2
5	Cryotherapy	90	6	2
6	Iris	150	4	3
7	Breast Cancer Wisconsin	569	30	2
8	Balance Scale	625	4	3
9	Dermatology	366	34	6
10	Haberman's Survival	306	3	2
11	Hepatitis	155	19	2
12	Teaching	151	5	3
13	Zoo	101	16	7
14	Libras	360	91	15
15	Hayes	132	5	3
16	Vehicle	846	18	4
17	Australian	690	14	2
18	*E. coli*	336	7	8
19	Musk (Version 2)	6598	168	2
20	Semeion	1593	256	2

3 Appreciating the performance of EigenClass in comparison with the well-known algorithms

The proposed EigenClass is compared with the well-known machine learning algorithms kNN, fuzzy kNN, RF, and MSVM through various datasets from UCI Machine Learning Repository [28] (Table 1) to assess their classification performance. The classification results are conducted via MATLAB R2018b running on a workstation with I(R) Xeon(R) CPU E5-1620 v4 @ 3.5 GHz, and 64 GB RAM. MATLAB source code of the proposed EigenClass algorithm is publicly available at: https://www.mathworks.com/matlabcentral/fileexchange/78462-eigenvalue-classification-for-machine-learning.

3.1 Comparative classification results

Each algorithm is trained and tested using the K-fold cross-validation [29]. In K-fold cross-validation, the dataset is split into K equal-sized subsamples whose samples are randomly designated. While one subsample is kept as validating data, the remaining K-1 subsamples are used in training the algorithm. In K-fold cross-validation, number of K processes are carried out for every subsample to be processed as validating data. Therefore, every subsample is employed only once as validating data, and the whole dataset is used for both training and validating data. In this study, 5-fold cross-validation is carried out and thus, the mean of the results for 5 iterations are recorded. Eventually, this process is repeated thirty times and their results are averaged to evaluate the performance measures. Meanwhile, some control parameters should be set for the compared algorithms. The number of the nearest neighbours and the distance function are considered as three and Euclidean distance for kNN, respectively. Similarly, the number of nearest neighbours is selected as 3 for fuzzy kNN. The Kernel function for MSVM is preferred as a radial basis function. The numbers of trees for the RF are set to be 60. Finally, k value used for calculation of k-mean is set to be 3 for the EigenClass algorithm.

The classification results of the algorithms are then compared among each other in terms of the most utilised measures, such as accuracy, precision, recall, micro-F-measure, and macro-F-measure [30] as defined below,

$$\text{Accuracy} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i} \tag{1}$$

$$\text{Precision} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{2}$$

$$\text{Recall} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{3}$$

$$\text{Macro - F measure} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{2 \times \text{TP}_i}{2 \times \text{TP}_i + \text{FP}_i + \text{FN}_i} \tag{4}$$

$$\text{Micro - F measure} = \frac{2\sum_{i=1}^{Q}\text{TP}_i}{2\sum_{i=1}^{Q}\text{TP}_i + \sum_{i=1}^{Q}\text{FP}_i + \sum_{i=1}^{Q}\text{FN}_i} \tag{5}$$

where TP_i, TN_i, FP_i, and FN_i are the number of true positives, true negatives, false positives, and false negatives for the label i, respectively, and Q is the total number of the class label.

Table 2 shows the classification results computed by kNN, Fuzzy kNN, MSVM, RF, and the proposed EigenClass algorithm through the 20 datasets presented in Table 1 concerning the measures, accuracy, precision, recall, micro-F-measure, and macro-F-measure with standard deviation (SD). The SD is an important metric that apprises the stability of an algorithm. It is calculated by the square root of the mean square difference between each solution and the average of the solutions found at each run. Since Table 2 includes large data, it is difficult to trace the results and interpret the performances of the algorithms. To facilitate this, Tables 3 and 4, in which the results of Table 2 are compiled by ranking the number of the best

Table 2 The classification results regarding the averages of accuracy, precision, recall, micro-F-measure, and macro-F-measure (%)

Dataset	Methods	Accuracy ± SD	Precision ± SD	Recall ± SD	Micro-F-measure ± SD	Macro-F-measure ± SD
Seeds	kNN	**95.68 ± 0.66**	**93.94 ± 0.90**	**93.52 ± 0.98**	**93.48 ± 0.98**	**93.52 ± 0.98**
	Fuzzy kNN	93.87 ± 0.95	91.28 ± 1.41	90.81 ± 1.42	90.74 ± 1.44	90.81 ± 1.42
	MSVM	94.25 ± 0.38	91.83 ± 0.67	91.38 ± 0.57	91.34 ± 0.60	91.38 ± 0.57
	RF	95.52 ± 0.53	93.80 ± 0.77	93.29 ± 0.79	93.26 ± 0.80	93.29 ± 0.79
	EigenClass	95.05 ± 0.01	92.86 ± 0.01	92.57 ± 0.01	92.51 ± 0.01	92.57 ± 0.01
Wireless	kNN	99.08 ± 0.07	98.19 ± 0.14	98.17 ± 0.14	98.17 ± 0.14	98.17 ± 0.14
	Fuzzy kNN	99.20 ± 0.07	98.42 ± 0.14	98.40 ± 0.15	98.40 ± 0.15	98.40 ± 0.15
	MSVM	98.95 ± 0.13	97.92 ± 0.26	97.89 ± 0.27	97.89 ± 0.27	97.89 ± 0.27
	RF	99.13 ± 0.10	98.28 ± 0.21	98.25 ± 0.21	98.25 ± 0.21	98.25 ± 0.21
	EigenClass	**99.37 ± 0.05**	**98.77 ± 0.04**	**98.75 ± 0.03**	**98.75 ± 0.07**	**98.75 ± 0.08**
Wine	kNN	96.93 ± 0.54	95.61 ± 0.76	96.15 ± 0.66	95.56 ± 0.80	95.39 ± 0.81
	Fuzzy kNN	83.33 ± 0.63	75.14 ± 1.13	74.13 ± 1.07	73.97 ± 1.07	74.99 ± 0.95
	MSVM	97.19 ± 0.51	95.97 ± 0.80	96.05 ± 0.75	95.83 ± 0.78	95.78 ± 0.77
	RF	98.43 ± 0.39	97.68 ± 0.52	97.91 ± 0.60	97.69 ± 0.57	97.64 ± 0.58
	EigenClass	**99.17 ± 0.01**	**98.85 ± 0.01**	**98.96 ± 0.01**	**98.85 ± 0.01**	**98.75 ± 0.01**
Immunotherapy	kNN	70.54 ± 2.16	58.16 ± 3.85	57.21 ± 4.56	63.02 ± 4.78	70.54 ± 2.16
	Fuzzy kNN	61.16 ± 3.13	44.84 ± 4.02	43.99 ± 3.47	64.17 ± 3.66	61.16 ± 3.13
	MSVM	79.17 ± 2.11	75.15 ± 5.05	55.22 ± 3.34	78.09 ± 2.75	79.17 ± 2.11
	RF	**84.94 ± 2.80**	**81.33 ± 5.96**	**71.29 ± 4.01**	**76.41 ± 4.19**	**84.94 ± 2.80**
	EigenClass	81.37 ± 0.02	81.14 ± 0.08	57.28 ± 0.03	77.80 ± 0.06	81.37 ± 0.02
Cryotherapy	kNN	88.58 ± 2.19	89.43 ± 2.23	88.34 ± 2.15	88.34 ± 2.23	88.58 ± 2.19
	Fuzzy kNN	90.04 ± 2.33	91.23 ± 1.99	89.93 ± 2.39	89.80 ± 2.45	90.04 ± 2.33
	MSVM	86.67 ± 2.98	87.94 ± 2.56	86.86 ± 3.14	86.49 ± 3.11	86.67 ± 2.98
	RF	91.04 ± 2.66	91.93 ± 2.07	91.23 ± 2.68	90.93 ± 2.80	91.04 ± 2.66
	EigenClass	**93.31 ± 0.04**	**94.20 ± 0.04**	**93.11 ± 0.05**	**93.17 ± 0.05**	**93.31 ± 0.04**
Iris	kNN	96.27 ± 0.52	94.74 ± 0.79	94.40 ± 0.78	94.38 ± 0.78	94.40 ± 0.78
	Fuzzy kNN	97.16 ± 0.37	96.01 ± 0.58	95.73 ± 0.56	95.72 ± 0.56	95.73 ± 0.56
	MSVM	97.51 ± 0.52	96.54 ± 0.82	96.27 ± 0.78	96.25 ± 0.78	96.27 ± 0.78
	RF	96.71 ± 0.52	95.45 ± 0.74	95.07 ± 0.78	95.05 ± 0.79	95.07 ± 0.78
	EigenClass	**97.98 ± 0.01**	**96.80 ± 0.01**	**96.47 ± 0.01**	**96.45 ± 0.01**	**96.47 ± 0.01**
Breast Cancer Wisconsin	kNN	95.06 ± 0.40	95.04 ± 0.42	94.43 ± 0.49	94.68 ± 0.44	95.06 ± 0.40
	Fuzzy kNN	91.39 ± 0.49	91.36 ± 0.62	90.28 ± 0.49	90.68 ± 0.53	91.39 ± 0.49
	MSVM	95.60 ± 0.48	95.68 ± 0.51	94.96 ± 0.54	95.26 ± 0.52	95.60 ± 0.48
	RF	96.01 ± 0.48	95.96 ± 0.56	95.57 ± 0.48	95.71 ± 0.51	96.01 ± 0.48
	EigenClass	**97.19 ± 0.01**	**96.86 ± 0.01**	**96.78 ± 0.01**	**96.75 ± 0.01**	**96.78 ± 0.01**
Balance Scale	kNN	85.56 ± 0.65	56.81 ± 0.38	56.67 ± 0.70	84.93 ± 0.75	78.34 ± 0.97
	Fuzzy kNN	85.84 ± 0.20	58.81 ± 0.34	56.98 ± 0.22	85.51 ± 0.30	78.77 ± 0.30
	MSVM	84.45 ± 0.01	83.28 ± 0.28	61.74 ± 0.04	85.65 ± 0.16	81.68 ± 0.01
	RF	89.44 ± 0.40	58.98 ± 0.32	60.88 ± 0.43	89.82 ± 0.45	84.16 ± 0.60
	EigenClass	**91.74 ± 0.01**	**87.88 ± 0.13**	**63.37 ± 0.02**	**92.10 ± 0.01**	**87.60 ± 0.03**
Dermatology	kNN	98.29 ± 0.25	94.80 ± 0.74	94.65 ± 0.84	94.46 ± 0.79	94.86 ± 0.76
	Fuzzy kNN	96.33 ± 0.33	87.99 ± 1.25	88.06 ± 0.97	87.42 ± 1.06	88.99 ± 0.98
	MSVM	97.24 ± 0.15	92.15 ± 0.53	89.24 ± 0.52	89.68 ± 0.58	91.73 ± 0.45
	RF	98.84 ± 0.16	96.36 ± 0.51	96.23 ± 0.66	96.04 ± 0.61	96.53 ± 0.49
	EigenClass	**99.20 ± 0.01**	**97.62 ± 0.02**	**97.14 ± 0.02**	**97.22 ± 0.02**	**97.60 ± 0.01**

Table 2 (continued)

Dataset	Methods	Accuracy ± SD	Precision ± SD	Recall ± SD	Micro-F-measure ± SD	Macro-F-measure ± SD
Haberman's Survival	kNN	65.07 ± 1.12	54.84 ± 1.45	54.92 ± 1.64	54.58 ± 1.52	65.07 ± 1.12
	Fuzzy kNN	68.20 ± 1.74	59.08 ± 2.54	58.98 ± 2.63	58.71 ± 2.42	68.20 ± 1.74
	MSVM	72.85 ± 0.88	63.16 ± 8.70	51.14 ± 1.39	62.21 ± 5.38	72.85 ± 0.88
	RF	68.36 ± 1.43	57.62 ± 2.26	56.21 ± 1.32	56.23 ± 1.68	68.36 ± 1.43
	EigenClass	**76.60 ± 0.03**	**70.06 ± 0.04**	**61.96 ± 0.02**	**63.67 ± 0.02**	**76.60 ± 0.03**
Hepatitis	kNN	57.29 ± 1.21	56.64 ± 1.70	56.38 ± 1.25	56.00 ± 1.49	57.29 ± 1.21
	Fuzzy kNN	57.61 ± 3.73	57.55 ± 4.03	57.41 ± 3.89	57.02 ± 3.85	57.61 ± 3.73
	MSVM	62.97 ± 2.47	63.11 ± 2.90	61.66 ± 2.50	60.99 ± 2.72	62.97 ± 2.47
	RF	62.00 ± 2.64	62.04 ± 2.93	61.24 ± 2.66	60.86 ± 2.68	62.00 ± 2.64
	EigenClass	**64.06 ± 0.02**	**64.73 ± 0.03**	**62.82 ± 0.03**	**62.29 ± 0.03**	**64.06 ± 0.03**
Teaching	kNN	74.33 ± 1.50	62.22 ± 2.31	61.59 ± 2.26	61.10 ± 2.06	61.49 ± 2.25
	Fuzzy kNN	72.27 ± 1.73	59.87 ± 3.18	58.33 ± 2.60	58.11 ± 2.85	58.41 ± 2.59
	MSVM	67.77 ± 1.45	53.69 ± 2.20	51.78 ± 2.20	51.06 ± 2.33	51.65 ± 2.18
	RF	75.38 ± 1.87	64.22 ± 3.03	62.96 ± 2.78	62.61 ± 2.81	63.07 ± 2.80
	EigenClass	**75.74 ± 0.02**	**64.96 ± 0.04**	**63.52 ± 0.03**	**63.12 ± 0.03**	**63.61 ± 0.03**
Zoo	kNN	97.57 ± 0.31	88.36 ± 2.73	83.58 ± 2.50	91.84 ± 1.10	91.67 ± 1.05
	Fuzzy kNN	98.87 ± 0.23	95.05 ± 1.25	90.26 ± 1.16	96.16 ± 1.73	96.16 ± 0.70
	MSVM	98.56 ± 0.46	92.57 ± 2.01	90.32 ± 2.76	94.99 ± 1.30	95.06 ± 1.59
	RF	98.57 ± 0.45	92.51 ± 2.39	88.00 ± 3.79	95.33 ± 1.56	95.12 ± 1.46
	EigenClass	**99.33 ± 0.46**	**96.53 ± 0.41**	**93.67 ± 0.35**	**97.87 ± 0.42**	**97.72 ± 0.42**
Libras	kNN	97.17 ± 0.09	81.97 ± 0.54	78.79 ± 0.67	78.43 ± 0.75	78.77 ± 0.68
	Fuzzy kNN	97.15 ± 0.18	81.81 ± 1.12	78.64 ± 1.27	78.29 ± 1.23	78.65 ± 1.33
	MSVM	93.37 ± 0.14	57.98 ± 1.67	50.47 ± 1.22	52.29 ± 1.08	50.25 ± 1.07
	RF	96.23 ± 0.28	75.27 ± 1.85	71.84 ± 2.04	71.87 ± 2.06	71.72 ± 2.08
	EigenClass	**97.52 ± 0.01**	**82.83 ± 0.06**	**79.07 ± 0.07**	**79.46 ± 0.07**	**79.39 ± 0.08**
Hayes	kNN	75.97 ± 2.99	71.10 ± 3.78	63.97 ± 4.22	65.52 ± 4.05	63.96 ± 4.48
	Fuzzy kNN	59.38 ± 2.13	37.44 ± 4.82	35.55 ± 2.94	39.12 ± 2.87	39.07 ± 3.20
	MSVM	74.38 ± 1.17	66.64 ± 1.77	62.63 ± 1.49	63.35 ± 1.53	61.57 ± 1.75
	RF	**87.42 ± 1.68**	**84.70 ± 2.14**	**83.62 ± 2.15**	**83.40 ± 2.30**	**81.13 ± 2.52**
	EigenClass	76.05 ± 0.02	69.39 ± 0.05	59.66 ± 0.05	60.28 ± 0.03	64.08 ± 0.04
Vehicle	kNN	84.45 ± 0.53	68.76 ± 1.10	69.09 ± 1.07	68.80 ± 1.07	68.90 ± 1.06
	Fuzzy kNN	81.91 ± 0.44	63.51 ± 0.86	64.17 ± 0.88	63.67 ± 0.85	63.82 ± 0.89
	MSVM	87.09 ± 0.44	73.30 ± 0.89	74.47 ± 0.88	73.55 ± 0.89	74.18 ± 0.89
	RF	**89.10 ± 0.23**	**78.43 ± 0.50**	**78.43 ± 0.47**	**78.24 ± 0.47**	**78.21 ± 0.47**
	EigenClass	83.31 ± 0.01	65.08 ± 0.01	66.85 ± 0.01	65.41 ± 0.01	66.63 ± 0.01
Australian	kNN	79.55 ± 0.84	79.49 ± 0.86	79.20 ± 0.85	79.23 ± 0.84	79.55 ± 0.84
	Fuzzy kNN	65.69 ± 0.58	65.29 ± 0.63	64.92 ± 0.59	64.92 ± 0.59	65.69 ± 0.58
	MSVM	75.08 ± 4.63	78.32 ± 5.36	73.94 ± 4.49	72.46 ± 5.48	.75.08 ± 4.63
	RF	83.01 ± 0.55	85.26 ± 0.53	81.55 ± 0.62	82.06 ± 0.58	83.01 ± 0.55
	EigenClass	**84.43 ± 0.01**	**86.51 ± 0.01**	**82.46 ± 0.01**	**82.95 ± 0.02**	**83.84 ± 0.01**
E. coli	kNN	94.41 ± 0.34	72.02 ± 2.42	67.76 ± 1.42	76.85 ± 1.66	80.13 ± 1.34
	Fuzzy kNN	94.60 ± 0.36	75.28 ± 2.12	71.22 ± 1.12	79.04 ± 1.35	81.10 ± 1.12
	MSVM	93.85 ± 0.18	77.61 ± 2.56	50.22 ± 1.57	76.56 ± 2.12	79.28 ± 0.55
	RF	94.69 ± 0.32	74.46 ± 1.96	71.25 ± 2.37	78.70 ± 1.47	81.01 ± 1.10
	EigenClass	**95.72 ± 0.28**	**83.53 ± 0.21**	**71.87 ± 0.20**	**82.64 ± 0.22**	**85.46 ± 0.24**

Table 2 (continued)

Dataset	Methods	Accuracy ± SD	Precision ± SD	Recall ± SD	Micro-F-measure ± SD	Macro-F-measure ± SD
Musk (Version 2)	kNN	94.75 ± 0.08	89.12 ± 0.21	91.48 ± 0.17	90.22 ± 0.15	94.75 ± 0.09
	Fuzzy kNN	95.32 ± 0.11	90.28 ± 0.32	92.38 ± 0.17	91.26 ± 0.18	95.32 ± 0.11
	MSVM	73.16 ± 3.51	60.49 ± 2.64	65.49 ± 3.78	60.73 ± 3.36	73.16 ± 3.51
	RF	96.21 ± 0.01	96.75 ± 0.03	92.01 ± 0.02	94.03 ± 0.08	96.18 ± 0.01
	EigenClass	**97.69 ± 0.11**	**98.00 ± 0.09**	**93.09 ± 0.34**	**95.33 ± 0.23**	**97.69 ± 0.10**
Semeion	kNN	**97.95 ± 0.24**	**97.18 ± 0.54**	**91.15 ± 1.04**	**93.80 ± 0.80**	**97.95 ± 0.25**
	Fuzzy kNN	97.44 ± 0.31	96.21 ± 0.62	89.13 ± 1.21	92.19 ± 1.04	97.44 ± 0.31
	MSVM	97.75 ± 0.21	94.79 ± 0.71	92.51 ± 1.39	93.52 ± 0.66	97.75 ± 0.21
	RF	96.36 ± 0.34	96.75 ± 0.45	82.49 ± 1.61	87.87 ± 1.33	96.36 ± 0.33
	EigenClass	96.72 ± 0.01	96.97 ± 0.01	90.99 ± 0.01	92.91 ± 0.01	96.74 ± 0.01

Table 3 Ranking number of the best results for all algorithm compared among each other

Methods	Accuracy	Precision	Recall	Micro-F-measure	Macro-F-measure	Total rank
kNN	2	2	1	2	2	9
Fuzzy kNN	–	–	–	–	–	–
MSVM	–	–	1	1	–	2
RF	3	3	3	2	3	14
EigenClass	15	15	15	15	15	75

Table 4 Ranking number of the best results for two algorithms compared versus each other

Comparison	Accuracy	Precision	Recall	Micro-F-measure	Macro-F-measure
EigenClass versus kNN	17	16	16	16	17
EigenClass versus Fuzzy kNN	18	20	20	20	19
EigenClass versus MSVM	17	19	17	16	18
EigenClass versus RF	16	16	16	17	16

results for every algorithm, are prepared. While Table 3 is composed of the ranking numbers of the best results for all the algorithms compared among each other, Table 4 includes the ranking numbers of the best results for two algorithms versus each other in pairwise comparisons. It can be understood from Table 3 that EigenClass outperforms the other algorithms for 15 datasets, while it produces close results (see also Table 2) for the remaining 5 datasets in view of each metric. Hence, EigenClass classifies datasets with the best results of the total number of 75 for all the measures. Conversely, the other algorithms kNN, Fuzzy kNN, MSVM, and RF classify the datasets with the total number of 9, 0, 2, and 14, respectively. Besides, it is clear from Table 4 that EigenClass outperforms the other algorithms for at least 16 datasets in every metric in the pairwise comparisons. Therefore, EigenClass is the most successful algorithm considering the overall classification performance. The comparison of the algorithms in terms of the SD yields that EigenClass is the most stable algorithms as well.

Note that the classification results in the literature show that a classification algorithm might not achieve the most successful results for all dataset. It can be inferred that the classification algorithms proposed in the literature might be problem-dependent, considering their classification abilities [31, 32].

3.2 Statistical evaluation

In this subsection, Friedman test [33] and the Nemenyi post hoc test [34] are exploited to assess whether the overall differences among the measures accuracy, precision, recall, micro-F-measure, and macro-F-measure are statistically significant. Friedman test is a nonparametric tool to test

multiple hypotheses and Nemenyi test is one of the post hoc tests widely used to compare the classifiers simultaneously. Friedman test ranks the algorithms based on their performances for each dataset separately; hence, the rank of 1 is assigned to the best performing algorithm, the rank of 2 to second-best, etc. It assigns average ranks in the case of obtaining an equal rank. Then, Friedman test compares the average ranks of the algorithms. Next, it calculates according to χ_F^2 distribution with $k-1$ degree of freedom (k is the number of algorithms). To detect a statistically significant difference in the performance, a post hoc test should be used to detect the algorithms which cause these differences. Nemenyi test is widely used for this case. This test shows that their performance is remarkably different in the event that the average ranks of the two algorithms are different from some critical distances.

We first calculate the average rank of each algorithm considered in our experiments with $k = 5$ and $n = 20$ since the total number of the methods is 5 and the total number of the datasets is 20. If the accuracy, precision, recall, micro-F-measure, and macro-F-measure values of the Friedman test statistic are $\chi_F^2 = 28.52$, $\chi_F^2 = 31.48$, $\chi_F^2 = 25.36$, $\chi_F^2 = 27.08$, and $\chi_F^2 = 31.08$, respectively, with $4(k-1)$ degrees of freedom and the critical value for the Friedman test given for $k = 5$ and $n = 20$ is 9.49 at a significance level of $\alpha = 0.05$, we can conclude that the accuracy, precision, recall, micro-F-measure, and macro-F-measure values of the studied methods are significantly different ($28.52 > 9.49$, $31.48 > 9.49$, $25.36 > 9.49$, $27.08 > 9.49$, and $31.08 > 9.49$, respectively). Now that the null hypothesis is rejected, we can proceed with a post hoc test. The Nemenyi test [33] can be used when all classifiers are compared to each other [35].

The critical value in our experiments with $k = 5$ and $\alpha = 0.05$ is $CD_{0.05} = 1.364$. As a result, the accuracy, precision, recall, micro-F-measure, and macro-F-measure of the proposed EigenClass method is significantly different from MSVM, kNN, and Fuzzy kNN methods, while it is not significantly different from RF method. Figure 1 illustrates the statistical comparison of the methods considered in our experiments based on the Nemenyi test.

3.3 Evaluation of processing time and computational complexity

To further compare the algorithms according to processing time, Table 5 provides the mean processing time data for the 20 UCI datasets at thirty runs. As can be inferred from Table 6, EigenClass, in general, seems to operate slightly slower than the other algorithms. The underlying cause of this higher processing time depends on the MATLAB operation. MATLAB does not allow for parallel computing for multiparameter eig(A, B) command set allows for only a single parameter, i.e. eig(A) [36]. We believe that the processing time can be further decreased if the parallel

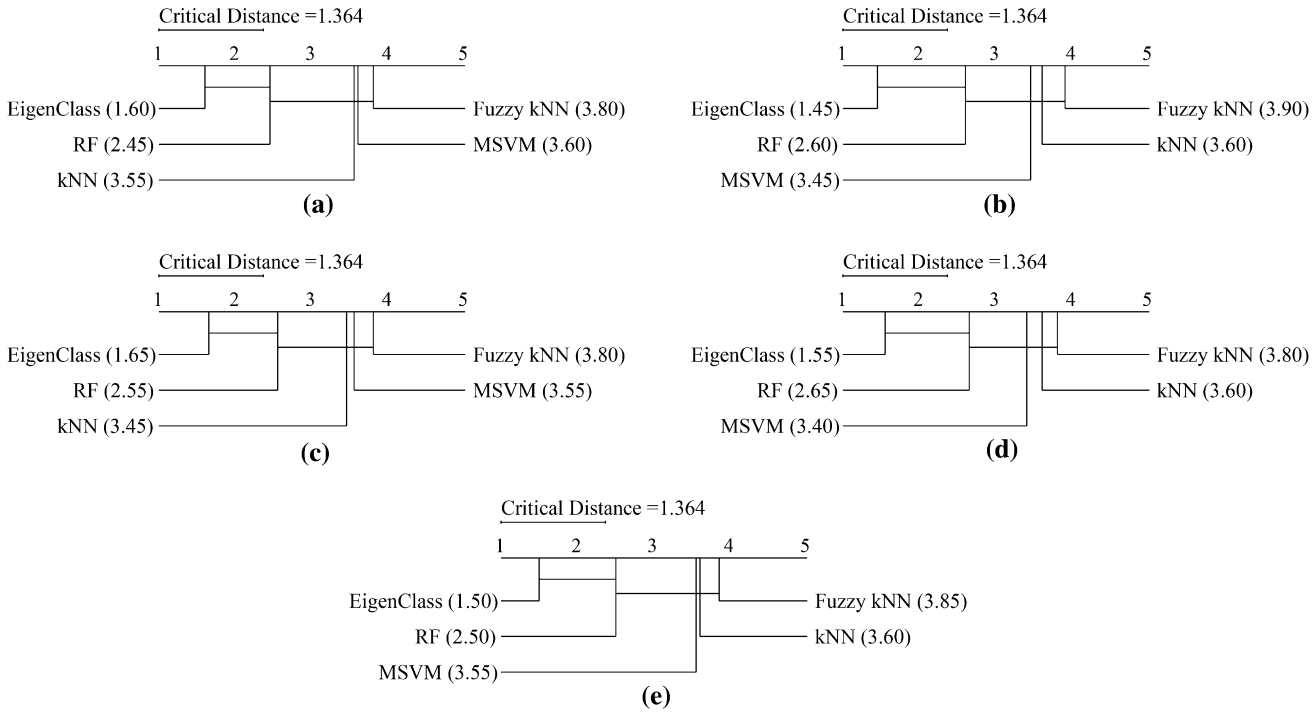

Fig. 1 The critical diagrams for the five measures [(a) Accuracy, (b) Precision, (c) Recall, (d) Micro-F-measure, (e) Macro-F-measure]: the results from the Nemenyi post hoc test at 0.05 significance level and average rank scores from Friedman Test

Table 5 Average processing time for the 20 UCI datasets (in seconds)

Dataset	kNN	Fuzzy KNN	MSVM	RF	EigenClass
Seeds	0.20	0.01	0.28	1.66	0.07
Wireless	0.32	0.04	1.40	13.32	5.34
Wine	0.05	0.01	0.61	1.23	0.06
Immunotherapy	0.02	0.01	0.50	0.66	0.01
Cryotherapy	0.02	0.01	0.32	0.65	0.01
Iris	0.03	0.01	0.09	1.05	0.02
Breast Cancer Wisconsin	0.10	0.01	4.53	3.80	1.23
Balance Scale	0.10	0.01	0.31	4.32	0.37
Dermatology	0.07	0.01	0.55	2.47	0.62
Haberman's survival	0.05	0.01	0.49	2.10	0.09
Hepatitis	0.03	0.01	1.45	1.09	0.06
Teaching	0.03	0.01	0.25	1.08	0.03
Zoo	0.02	0.01	0.27	0.74	0.02
Libras	0.06	0.01	2.86	2.71	0.74
Hayes	0.03	0.01	0.14	0.96	0.03
Vehicle	0.14	0.01	7.62	6.12	1.44
Australian	0.12	0.01	7.41	4.75	0.91
E. coli	0.05	0.01	0.81	2.36	0.13
Musk (Version 2)	8.00	3.50	98.90	47.30	153.70
Semeion	0.78	0.28	0.56	10.81	59.36

Table 6 Computational complexity of the algorithms

Method	Computational complexity
kNN [39]	$O(n \log k)$
Fuzzy kNN [40]	$O(n^2 \log(k))$
MSVM with Kernel [41]	$O(m^3)$
RF [42]	$O(Mmn \log n)$
EigenClass	$O(m \times n)$

k, number of neighbours; M, number of trees

computing property is activated for the multiparameter command and by optimising the coding of the algorithm in future works. Nevertheless, EigenClass computes the classification tasks under 1 s for 15 datasets. The processing time of an algorithm should be independently evaluated in consideration of the computational complexity based on big O notation [37, 38]. From the pseudocode of the EigenClass algorithm, the computational complexities are $O(m \times n)$ for Step 2, $O(n \times l)$ for Steps 3–4, and $O(n)$ for Steps 5–7 for each, where m and n are the numbers of attributes and samples, respectively. Because $m \times n$ is higher than $n \times l$, the general computational complexity is $O(m \times n)$ in terms of big O notation. The computational complexities of the compared algorithms are given in Table 6. Therefore, EigenClass has the second lowest computational complexity together with kNN after Fuzzy kNN.

In conclusion, the developed EigenClass algorithm is generally superior to the other existing well-known algorithms in view of the accuracy, precision, recall, micro-F-measure, and macro-F-measure measures. Moreover, it seems that EigenClass suffers from two minor disadvantages. The one is, as mentioned above, to assign a very small value (i.e. 0.0001) instead of 0 (zero) entries, resulting in a modification of the original dataset that may affect the performance of the algorithm. That is why EigenClass is not as successful for Hayes and Semeion datasets as the other algorithms. Note that Semeion dataset particularly consists of binary data. The second is related to the processing time that tends to increase when the number of samples and attributes of the dataset grows higher.

4 Conclusion

In the present study, a novel machine learning algorithm, namely EigenClass, which is built on the concept of eigenvalue, has been proposed for the classification problems. EigenClass is based on eigenvalue's notable proximity evaluation capability. Classification performance of EigenClass is evaluated through an effective comparison with several proven machine learning algorithms, i.e. kNN, fuzzy kNN, RF, and MSVM. The algorithms are trained and tested for 30 runs using 5-fold cross-validation over the

most frequently utilised 20 datasets. The results are then evaluated using several well-known measures, such as accuracy, precision, recall, micro-F-measure, and macro-F-measure. The EigenClass algorithm more accurately classifies most of the evaluated datasets. To exemplify, it yields the best results in the occurrence of 15 datasets in terms of every metric when each is compared with the other and in at least 16 datasets for each metric in a pairwise comparison. Statistical analyses and computational complexity are performed to further examine the precision and processing time of the algorithms by comparing them with the others. Hence, EigenClass generally outperforms the algorithms in the comparison in view of the results. Therefore, this study is the first to develop an eigenvalue-based machine learning algorithm, which is accurate and efficient. The achieved results intended for some classification problems are sufficient to prove the success of EigenClass though it can be further improved by optimising the steps and coding of the algorithm.

Compliance with ethical standards

Conflict of interest The author declares that they have no conflict of interest.

References

1. Jin R, Zhang J (2007) Multi-class learning by smoothed boosting. Mach Learn 67:207–227. https://doi.org/10.1007/s10994-007-5005-y
2. Takenouchi T, Ishii S (2018) Binary classifiers ensemble based on Bregman divergence for multi-class classification. Neurocomputing 273:424–434. https://doi.org/10.1016/j.neucom.2017.08.004
3. Li P (2019) Research on radar signal recognition based on automatic machine learning. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04494-1
4. Takenouchi T, Ishii S (2011) Ternary Bradley-Terry model-based decoding for multi-class classification and its extensions. Mach Learn 85:249–272. https://doi.org/10.1007/s10994-011-5240-0
5. Xu H, Wang W, Qian Y (2017) Fusing complete monotonic decision trees. IEEE Trans Knowl Data Eng 29:2223–2235. https://doi.org/10.1109/TKDE.2017.2725832
6. Liu T, Tao D (2016) Classification with noisy labels by importance reweighting. IEEE Trans Pattern Anal Mach Intell 38:447–461. https://doi.org/10.1109/TPAMI.2015.2456899
7. Langseth H, Nielsen TD (2006) Classification using hierarchical Naïve Bayes models. Mach Learn 63:135–159. https://doi.org/10.1007/s10994-006-6136-2
8. Nebel D, Kaden M, Villmann A, Villmann T (2017) Types of (dis-)similarities and adaptive mixtures thereof for improved classification learning. Neurocomputing 268:42–54. https://doi.org/10.1016/j.neucom.2016.12.091
9. Quost B, Destercke S (2017) Classification by pairwise coupling of imprecise probabilities. Pattern Recognit 77:412–425. https://doi.org/10.1016/j.patcog.2017.10.019
10. Law A, Ghosh A (2019) Multi-label classification using a cascade of stacked autoencoder and extreme learning machines. Neurocomputing 358:222–234. https://doi.org/10.1016/j.neucom.2019.05.051
11. Samaniego L, Bárdossy A, Schulz K (2008) Supervised classification of remotely sensed imagery using a modified k-NN technique. IEEE Trans Geosci Remote Sens 46:1–26. https://doi.org/10.1109/TGRS.2008.916629
12. Warfield S (1996) Fast k-NN classification for multichannel image data. Pattern Recognit Lett 17:713–721. https://doi.org/10.1016/0167-8655(96)00036-0
13. Zhang JJ, Fang M, Li X (2017) Clustered intrinsic label correlations for multi-label classification. Expert Syst Appl 81:134–146. https://doi.org/10.1016/j.eswa.2017.03.054
14. Liu Z, Cheng Y, Wang P et al (2018) A method for remaining useful life prediction of crystal oscillators using the Bayesian approach and extreme learning machine under uncertainty. Neurocomputing 305:27–38. https://doi.org/10.1016/j.neucom.2018.04.043
15. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297. https://doi.org/10.1023/A:1022627411411
16. Dudani SA (1976) The distance-weighted k-Nearest-neighbor rule. IEEE Trans Syst Man Cybern SMC-6:325–327. https://doi.org/10.1109/tsmc.1976.5408784
17. Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
18. Li S, Song S, Wan Y (2018) Laplacian twin extreme learning machine for semi-supervised classification. Neurocomputing 321:17–27. https://doi.org/10.1016/j.neucom.2018.08.028
19. Zhang C, Liu C, Zhang X, Almpanidis G (2017) An up-to-date comparison of state-of-the-art classification algorithms. Expert Syst Appl 82:128–150. https://doi.org/10.1016/j.eswa.2017.04.003
20. Noh Y-K, Zhang B-T, Lee DD (2018) Generative local metric learning for nearest neighbor classification. IEEE Trans Pattern Anal Mach Intell 40:106–118. https://doi.org/10.1109/TPAMI.2017.2666151
21. Wang X, Shen S, Shi G et al (2016) Iterative non-local means filter for salt and pepper noise removal. J Vis Commun Image Represent 38:440–450. https://doi.org/10.1016/j.jvcir.2016.03.024
22. Vladimir Naumovich V (1998) Statistical learning theory. Springer, New York
23. Ai Q, Wang A, Wang Y, Sun H (2019) An improved Twin-KSVC with its applications. Neural Comput Appl 31:6615–6624. https://doi.org/10.1007/s00521-018-3487-0
24. Jolliffe IT (2002) Principal component analysis, 2nd edn. Springer, New York
25. Balakrishnama S, Ganapathiraju A (1998) Linear discriminant analysis—a brief tutorial. Inst Signal Inf Process 18:1–8
26. Keller JM, Gray MR (1985) A fuzzy K-nearest neighbor algorithm. IEEE Trans Syst Man Cybern SMC-15:580–585. https://doi.org/10.1109/tsmc.1985.6313426
27. Shultz TR, Mareschal D, Schmidt WC (1994) Modeling cognitive development on balance scale phenomena. Mach Learn. https://doi.org/10.1023/A:1022630902151
28. Dua D, Graff C (2019) UCI machine learning repository. School of Information and Computer Sciences University of California. http://archive.ics.uci.edu/ml. Accessed 13 Aug 2019
29. Ustun D, Toktas A, Akdagli A (2019) Deep neural network-based soft computing the resonant frequency of E-shaped patch antennas. AEU Int J Electron Commun 102:54–61. https://doi.org/10.1016/j.aeue.2019.02.011
30. Nguyen TT, Dang MT, Luong AV et al (2019) Multi-label classification via incremental clustering on an evolving data

stream. Pattern Recognit 95:96–113. https://doi.org/10.1016/j.patcog.2019.06.001
31. Abdar M, Zomorodi-Moghadam M, Zhou X et al (2018) A new nested ensemble technique for automated diagnosis of breast cancer. Pattern Recognit Lett. https://doi.org/10.1016/j.patrec.2018.11.004
32. Adem K, Kiliçarslan S, Cömert O (2019) Classification and diagnosis of cervical cancer with softmax classification with stacked autoencoder. Expert Syst Appl 115:557–564. https://doi.org/10.1016/j.eswa.2018.08.050
33. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11:86–92. https://doi.org/10.1214/aoms/1177731944
34. Nemenyi P (1963) Distribution-free multiple comparisons. Ph.D. Princeton University
35. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
36. Eigenvalues and eigenvectors. https://www.mathworks.com/help/matlab/ref/eig.html. Accessed 17 Feb 2020
37. Bachmann P (1894) Analytische Zahlentheorie, vol 2. Teubner, Leipzig **(in German)**
38. Landau E (1909) Handbuch der Lehre von der Verteilung der Primzahlen. B. G. Teubner, Leipzig **(in German)**
39. Maillo J, Luengo J, García S et al (2017) Exact fuzzy k-nearest neighbor classification for big datasets. In: IEEE international conference on fuzzy systems (FUZZ-IEEE)
40. Nikdel H, Forghani Y, Mohammad Hosein Moattar S (2018) Increasing the speed of fuzzy k-nearest neighbours algorithm. Expert Syst 35:e12254. https://doi.org/10.1111/exsy.12254
41. Tsang IWH, Kwok JTY, Zurada JM (2006) Generalized core vector machines. IEEE Trans Neural Netw 17:1126–1140. https://doi.org/10.1109/TNN.2006.878123
42. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18:1153–1176. https://doi.org/10.1109/COMST.2015.2494502