Arabian Journal for Science and Engineering (2020) 45:10621–10635
https://doi.org/10.1007/s13369-020-04788-w

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Two Pseudo-Common Vectors for Pattern Recognition

Mehmet Koc[1] · M. Bilginer Gülmezoğlu[2] · Semih Ergin[2] · Rifat Edizkan[2] · Atalay Barkana[3]

Received: 13 September 2019 / Accepted: 12 July 2020 / Published online: 3 August 2020

Abstract
In this paper, the mathematical model used in finding the common vectors of classes in pattern recognition problems is reconsidered to obtain possible alternative solutions for the common vectors. Since the number of unknowns is always one larger than the number of equations in the mathematical model, the best solution to the problem seems to be the pseudo-inverse solutions. We obtained two forms of common vectors, called "pseudo-common vectors," using the proposed idea. Computational simplifications are accomplished as shown in the paper since we know that taking pseudo-inverses is an exhaustive procedure especially in the high-dimensional vector spaces. The two forms of pseudo-common vectors obtained in the paper are used in the classification of the data given in TI-Digit, AR-Face, and MNIST databases separately in order to see their effectiveness.

Keywords Pseudo-common vector · Pseudo-inverse · Common vector · Pattern recognition

1 Introduction

Image processing and machine vision is a technique of the electronics field useful for a human action recognition system. In order to develop a system with high performance, the effect of scaling, rotation, occlusion, noise, and other factors must be handled properly. Different types of noise are additive or multiplicative. Accordingly, the additive noise model is not adequate for complicated image acquisition processes such as synthetic aperture radar (SAR) imaging. Furthermore, speckle noise severely damages an image because of its multiplicative property. Speckle noise is currently the most important issue in image processing, and many researchers have been working on filter design to remove it [1, 2].

Machine vision is an interesting application of image processing, useful for developing electronic systems, in which machine intelligence is mainly used. There is a variety of machine vision branches, and researchers have great interest in many modern pattern recognition algorithms [2–6]. In image analysis, one of the important tasks is image segmentation, where the image is divided into some categories based on some criteria. There are several image segmentation techniques that have been proposed in the literature such as thresholding, region-based, edge detection, clustering, and weakly supervised learning in CNN. [1, 4, 6, 7].

In recent years, deep learning became one of the most preferred machine learning methods in many areas. Deep learning uses a network structure consisting of multiple layers. As such, convolutional neural network (CNN) is a deep learning algorithm categorized as supervised learning. In a recent study, a hybrid CNN-MLP classifier was proposed as an efficient method in the pattern classification problem [8]. Moreover, deep belief network (DBN) is an unsupervised learning algorithm constructed by stacking multiple restricted Boltzmann machines (RBM) [1]. DBN can be used to map the set of features to another set

✉ Mehmet Koc
mehmet.koc@bilecik.edu.tr

M. Bilginer Gülmezoğlu
bgulmez@ogu.edu.tr

Semih Ergin
sergin@ogu.edu.tr

Rifat Edizkan
redizkan@ogu.edu.tr

Atalay Barkana
atalaybarkan@anadolu.edu.tr

[1] Department of Electrical and Electronics Engineering, Bilecik Seyh Edebali University, Bilecik, Turkey

[2] Department of Electrical and Electronics Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey

[3] Department of Electrical and Electronics Engineering, Eskisehir Technical University, Eskisehir, Turkey

of features representing the classes better than the original set of features.

There are new image processing techniques and many machine vision applications such as texture recognition [9, 10], coastline detection [11, 12], and iris tissue recognition [13]. Raeisi et al. [14] developed a new algorithm to discriminate between oil spills and lookalikes. The testing of the developed algorithm on the several real SAR images provided high performance for both dark spot detection and oil spill classification. In another machine vision application, robust reservoir rock fracture recognition based on a new sparse feature learning and data training method was realized [15]. The algorithm proposed that paper identifies the sine fractures of reservoir rock automatically and is highly capable of determining the fractures of the imaging logs successfully.

On the other hand, the motivation for introducing this work is based on our early works discovered and published in IEEE Transactions on Speech and Audio Processing [16, 17]. We faced a problem with isolated word recognition on a database containing 20 Turkish words repeated by about 200 persons of different genders and ages [16]. Our primary problem in the recognition stage of isolated words was the intra-speaker differences along with the inter-speaker differences for words of the same class. At the time, we believed that we could eliminate all possible differences in each word repeated by about 200 persons. Therefore, we started with a mathematical model representing differences in the pronunciations, the common part of the repetitions in the same class, and the possible errors in the calculations and recordings. In other words, for repetitions belonging to one class or word, we had the following:

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{a}_{\text{com}} + \boldsymbol{b}_1 + \boldsymbol{\varepsilon}_1 \\ \boldsymbol{a}_2 &= \boldsymbol{a}_{\text{com}} + \boldsymbol{b}_2 + \boldsymbol{\varepsilon}_2 \\ &\vdots \\ \boldsymbol{a}_m &= \boldsymbol{a}_{\text{com}} + \boldsymbol{b}_m + \boldsymbol{\varepsilon}_m \end{aligned} \tag{1}$$

where $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m$ are the vectors, more frequently known as feature vectors ($n \times 1$) in the pattern recognition area; m is the number of feature vectors used in the training stage, $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m$ are the difference vectors representing the inter-class and intra-class differences; $\boldsymbol{a}_{\text{com}}$ is called the common vector, which includes invariant properties of each class used in the training stage, and $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_m$ are the error vectors that may arise in calculations. We assume that $\boldsymbol{a}_{\text{com}}$ represents the common part of one of the classes in the training set.

Now, the main problem after setting up a mathematical model for the repetitions was to determine the common vectors, $\boldsymbol{a}_{\text{com}}$'s, of all classes. In fact, when all vectors in class $C = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m\}$ (m is also the number of vector equations) in the training set were given, we had to calculate $m+1$ vectors, that is, one common vector and m difference vectors. In other words, we had m vector equations and $m+1$ unknown vectors. This is known as the underdetermined case. Furthermore, the number of unknown vectors is always one larger than the number of vector equations in the mathematical model. Unfortunately, there are infinite solutions of unknown $m+1$ vectors that may satisfy the mathematical model; in other words, the solution of the unknown vectors is not ever unique. This problem was solved previously using subspace methods when the number of vectors used in the training set was less than or equal to the vector dimension [16, 17]. This was known as the insufficient data case. Moreover, each class had its own difference subspace, that is, the different classes had different difference subspaces. Later on, a unique difference subspace was formed by combining all difference subspaces of all the classes to be recognized. In addition, it was used in the calculation of the common vectors of all the classes, which is known as a small sample size problem in the literature of pattern recognition area. The common vectors in this final case were called the discriminative common vectors [18].

In this study, we propose a pseudo-inverse-based approach to calculate the common vectors, which will be unique for each class and referred to as "pseudo-common vectors" hereon. With this new idea, we will demonstrate the possibility of other solutions for the common vectors. For calculating pseudo-common vectors for each class, we must introduce a couple of constraints before trying to solve the relations in the mathematical model given in Eq. (1). One constraint will be that the common vector, $\boldsymbol{a}_{\text{com}}$, should never be a zero vector and must be unique. The error vectors in the model may be combined with the difference vectors in the relations. However, in any case, they should not be too large in magnitude or any larger than their training vectors. This leads us to the use of pseudo-inverses to assign reasonable values to the common and the difference vectors. In the experimental studies, the performance of the classifier is tested in three different databases: one for isolated speech recognition, one for face recognition, and the other for handwritten character recognition. It must be pointed out that the proposed approach is simpler and faster when compared to the calculation of pseudo-inverses, which will be explained in the rest of the paper.

In the next section, we will introduce some basic knowledge about pseudo-inverses. Section 3 includes the computation of the first form of a pseudo-common vector. The simplification of the calculations is described in Sect. 4. Furthermore, the computation of the second form of pseudo-common vector is presented in Sect. 5. Experimental work

on three databases for classification is given in Sect. 6, while Sects. 7 and 8 include the discussion and conclusion, respectively.

2 Background for the Pseudo-Inverse

In many machine learning algorithms, it is necessary to find a solution to linear equations of the type $\boldsymbol{Ax} = \boldsymbol{y}$ with m equations and n unknowns for the underdetermined case ($m < n$). Since there are more unknowns than equations, infinitely many solutions are available. In this case, least squares can be used to choose a solution $\boldsymbol{x}'$ among the many solutions that minimize $\boldsymbol{Ax}' - \boldsymbol{y}$. If the matrix $\boldsymbol{A}$ has full rank, i.e., rank($\boldsymbol{A}$) = m, then the pseudo-inverse of $\boldsymbol{A}$ is obtained from $\boldsymbol{A}^{+} = \boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}\right)^{-1}$ (especially when $m > n$) [19]. However, if $\boldsymbol{A}$ is rank-deficient, the Moore–Penrose inverse of $\boldsymbol{A}$ cannot be determined from $\boldsymbol{A}^{+} = \left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathrm{T}}$. In this case, singular value decomposition (SVD) can be used to calculate $\boldsymbol{A}^{+}$. The solution for least square problems can be obtained by solving the normal equations or using QR decomposition or SVD [20, 21]. The SVD approach generally is slower than QR, but both of them are stable [20]. Many algorithms have been proposed for speeding up the computation of the pseudo-inverse [22–24]. Furthermore, the pseudo-inverse has been used in many applications such as regression, data analysis, and classification [25–29].

Pseudo-inverses are one of the subtle subjects for engineering students, mainly used to solve $\boldsymbol{x} \in R^{n\times 1}$ in the linear system of equations as mentioned in the previous paragraph:

$$\boldsymbol{Ax} = \boldsymbol{y} \tag{2}$$

where $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{y}$ must be an $m \times 1$ vector and both are well defined numerically. The vector, $\boldsymbol{x}$, has a size of $n \times 1$ whose components are to be determined. This relation is mainly used for linear regression (or curve fitting) problems. Let us start with a simple example to clarify our point.

Example Let us try to find the best line equation that passes as close as possible to the three data points with coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ in a two-dimensional space. The line equation in this case may be expressed with

$$y = \alpha x + \beta \tag{3}$$

where α is the slope and β is the y-intercept of the line. The parameters α and β are to be determined. A vector–matrix relation can easily be established using the idea that each of the three data points must approximately satisfy the line equation above. That is,

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \tag{4}$$

or else

$$\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{\varepsilon}, \quad \boldsymbol{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{5}$$

where the components of the 3×1 vector $\boldsymbol{y}$ are the y-coordinates of the points, α and β are the components of the 2×1 vector $\boldsymbol{x}$; $\boldsymbol{\varepsilon}$ is the 3×1 vector of errors, and $\boldsymbol{A}$ is the only matrix in the relation which has the size of 3×2. The components of the vector $\boldsymbol{x}$ can be calculated by minimizing the sum of error squares, that is, we will start with the error vector:

$$\boldsymbol{\varepsilon} = \boldsymbol{y} - \boldsymbol{Ax}$$

then, the sum of the error squares is $\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 = \left(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\right)(\boldsymbol{y} - \boldsymbol{Ax})$.

In order to minimize the sum of error squares with respect to the vector $\boldsymbol{x}$, the critical point must be calculated by taking its derivative and equating the result to the zero vector:

$$\frac{\mathrm{d}\left(\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}\right)}{\mathrm{d}\boldsymbol{x}} = 0 = 2\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ax} - 2\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}$$

If the inverse of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ exists, then

$$\boldsymbol{x} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \tag{6}$$

This is the overdetermined case, and the solution is usually possible when the number of data points is equal to or larger than the number of components in the vector $\boldsymbol{x}$. When the number of data points is equal to the number of components in $\boldsymbol{x}$, a direct solution is possible as

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y} \tag{7}$$

whenever $\boldsymbol{A}^{-1}$ exists. However, when the number of data points is less than the number of components in $\boldsymbol{x}$, then there are infinitely many possible solutions for $\boldsymbol{x}$ in $\boldsymbol{Ax} = \boldsymbol{y}$. This is called the underdetermined case, where if one is still looking for a unique solution, some constraints must be imposed on the vector $\boldsymbol{x}$.

It could also be noted that the parameters in the linear set of equations may be linearly dependent. However, our problem is that we should not introduce these sorts of complications.

Therefore, we will instead try to find a unique solution to $\boldsymbol{Ax} = \boldsymbol{y}$, e.g., under the constraint, "$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$ must have a minimum value while solving $\boldsymbol{x}$ in $\boldsymbol{Ax} = \boldsymbol{y}$." This will lead us to a unique solution with the pseudo-inverse concept. One of the best descriptions of pseudo-inverses is given in [30].

In summary, if $\boldsymbol{A}^{-1}$ does not exist in relation to $\boldsymbol{Ax} = \boldsymbol{y}$, then the pseudo-inverse of $\boldsymbol{A}$, which is shown with $\boldsymbol{A}^{+}$, can be used for calculating the vector $\boldsymbol{x}$ as

$$\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{y}. \tag{8}$$

3 Computation of the First Form of the Pseudo-Common Vector

The application of pseudo-inverses will be shown for a one-class problem only to avoid any ambiguity. The reader can easily extend the idea to any multi-class problem. Suppose that we have a class of feature vectors (by definition, feature vectors are vectors whose components are numerical values assigned to the features of the objects under classification),

$$C = \left\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m\right\} \tag{9}$$

with the mathematical model given in Eq. (1), which will be repeated in here by replacing $\boldsymbol{a}_{\mathrm{com}}$ with $\boldsymbol{a}_{\mathrm{pcom}}$ for convenience

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{a}_{\mathrm{pcom}} + \boldsymbol{b}_1 + \boldsymbol{\varepsilon}_1 \\ \boldsymbol{a}_2 &= \boldsymbol{a}_{\mathrm{pcom}} + \boldsymbol{b}_2 + \boldsymbol{\varepsilon}_2 \\ &\vdots \\ \boldsymbol{a}_m &= \boldsymbol{a}_{\mathrm{pcom}} + \boldsymbol{b}_m + \boldsymbol{\varepsilon}_m \end{aligned} \tag{10}$$

where $\boldsymbol{a}_i, \boldsymbol{a}_{\mathrm{pcom}}, \boldsymbol{b}_i, \boldsymbol{\varepsilon}_i \in R^{n\times 1}$ for $i = 1, 2, \ldots, m$. We preferred to use the newly introduced pseudo-common vector as $\boldsymbol{a}_{\mathrm{pcom}}$, in order not to confuse it with the previously calculated common vector $\boldsymbol{a}_{\mathrm{com}}$. The above set of equations can be combined and represented with one vector–matrix equation:

$$\begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{n\times n} & \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \cdots & \boldsymbol{0}_{n\times n} \\ \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \boldsymbol{I}_{n\times n} & \cdots & \boldsymbol{0}_{n\times n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \boldsymbol{0}_{n\times n} & \cdots & \boldsymbol{I}_{n\times n} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{\mathrm{pcom}} \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_m \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \vdots \\ \boldsymbol{\varepsilon}_m \end{bmatrix} \tag{11}$$

where the first row of Eq. (11) corresponds to the first vector equation in Eq. (10), the second row of Eq. (11) corresponds to the second vector equation in Eq. (10), and so on. Or, more simply, define the vector $\boldsymbol{y}$ as $\boldsymbol{y}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{a}_1^{\mathrm{T}} & \boldsymbol{a}_2^{\mathrm{T}} & \ldots & \boldsymbol{a}_m^{\mathrm{T}} \end{bmatrix} \in R^{1\times mn}$, the parameter vector to be identified $\boldsymbol{x}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{a}_{\mathrm{pcom}}^{\mathrm{T}} & \boldsymbol{b}_1^{\mathrm{T}} & \boldsymbol{b}_2^{\mathrm{T}} & \ldots & \boldsymbol{b}_m^{\mathrm{T}} \end{bmatrix} \in R^{1\times(m+1)n}$, and the error vector $\boldsymbol{\varepsilon}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{\varepsilon}_1^{\mathrm{T}} & \boldsymbol{\varepsilon}_2^{\mathrm{T}} & \ldots & \boldsymbol{\varepsilon}_m^{\mathrm{T}} \end{bmatrix} \in R^{1\times mn}$. The only matrix $\boldsymbol{A}$ in the above relation is $mn \times (m+1)n$ dimensional. Finally, the vector–matrix equation can be written as:

$$\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{\varepsilon}$$

The vector $\boldsymbol{x}$ can be solved using the aforementioned pseudo-inverse concept as $\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{y}$ which minimizes the sum of error squares. A direct use of the vector–matrix relations may cause problems in using the MATLAB type of programs due to the extremely high dimensionality of the matrix $\boldsymbol{A}$ in most of the classification problems.

Similarly, when a feature vector, $\boldsymbol{a}_x$, whose class is unknown in the test set is taken to determine its class, a similar procedure must be followed. The math model with the inclusion of $\boldsymbol{a}_x$ will be as follows:

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{a}'_{\mathrm{pcom}} + \boldsymbol{b}'_1 + \boldsymbol{\varepsilon}'_1 \\ &\vdots \\ \boldsymbol{a}_m &= \boldsymbol{a}'_{\mathrm{pcom}} + \boldsymbol{b}'_m + \boldsymbol{\varepsilon}'_m \\ \boldsymbol{a}_x &= \boldsymbol{a}'_{\mathrm{pcom}} + \boldsymbol{b}'_x + \boldsymbol{\varepsilon}'_x \end{aligned} \tag{12}$$

Or with the vector–matrix relation

$$\begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_m \\ \boldsymbol{a}_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{n\times n} & \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \cdots & \boldsymbol{0}_{n\times n} \\ \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \boldsymbol{I}_{n\times n} & \cdots & \boldsymbol{0}_{n\times n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I}_{n\times n} & \boldsymbol{0}_{n\times n} & \boldsymbol{0}_{n\times n} & \cdots & \boldsymbol{I}_{n\times n} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}'_{\mathrm{pcom}} \\ \boldsymbol{b}'_1 \\ \boldsymbol{b}'_2 \\ \vdots \\ \boldsymbol{b}'_m \\ \boldsymbol{b}'_x \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon}'_1 \\ \boldsymbol{\varepsilon}'_2 \\ \vdots \\ \boldsymbol{\varepsilon}'_m \\ \boldsymbol{\varepsilon}'_x \end{bmatrix} \tag{13}$$

where $\boldsymbol{y}'$ is the vector on the left-hand side of equality with $(m+1)n \times 1$ dimensions, $\boldsymbol{x}'$ is the parameter vector to be calculated using pseudo-inverse solution with $(m+2)n \times 1$ dimensions, $\boldsymbol{A}'$ is the matrix in front of $\boldsymbol{x}'$ with $(m+1)n \times (m+2)n$ dimensions, and $\boldsymbol{\varepsilon}'$ is the error vector with $(m+1)n \times 1$ dimensions. The vector–matrix relation in this case will be

$$\boldsymbol{y}' = \boldsymbol{A}'\boldsymbol{x}' + \boldsymbol{\varepsilon}' \tag{14}$$

The pseudo-inverse solution of $\boldsymbol{x}'$ will be

$$\boldsymbol{x}' = \left(\boldsymbol{A}'\right)^{+}\boldsymbol{y}' \tag{15}$$

This minimizes the sum of error squares.

The distance of the test vector $\boldsymbol{a}_x$ to the pseudo-common vector of that class can be measured from

$$d_x^2 = \left\|\boldsymbol{a}'_{\mathrm{pcom}} - \boldsymbol{a}_{\mathrm{pcom}}\right\|^2 \tag{16}$$

This will help us in the classification process of $\boldsymbol{a}_x$ and will be used in the decision criterion. If $j = 1, 2, \ldots, c$ indicates the class numbers in $C = \left\{C_1, C_2, \ldots, C_c\right\}$, then the decision criterion will be

$$C_x = \underset{x=j}{\operatorname{argmin}} \left\|\boldsymbol{a}'^{j}_{\mathrm{pcom}} - \boldsymbol{a}^{j}_{\mathrm{pcom}}\right\|^2 \quad \text{for} \quad j = 1, 2, \ldots, c. \tag{17}$$

If the distance is minimum for cth class, $\boldsymbol{a}_x$ is assigned to that class.

Let us remember that the operations with the vector–matrix relations will be troublesome. Therefore, they need simplifications at least in their own dimensionality for the sake of calculability.

4 Simplification of the Calculations

4.1 Conversion of the Vector–Matrix Relations Back to Scalar Relations

Extreme dimensionality in $\boldsymbol{A}$ and $\boldsymbol{A}'$ matrices is not important at all because the vector equations of the type in Eq. (1) can be converted into their scalar components with ease. Let us explain this with a simple example.

Example Suppose that the class C has two feature vectors in its training set, then $C = \{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ with its math model

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{a}_{\text{pcom}} + \boldsymbol{b}_1 + \boldsymbol{\varepsilon}_1 \\ \boldsymbol{a}_2 &= \boldsymbol{a}_{\text{pcom}} + \boldsymbol{b}_2 + \boldsymbol{\varepsilon}_2 \end{aligned} \tag{18}$$

Also, suppose that $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_{\text{pcom}}$, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{\varepsilon}_1$, and $\boldsymbol{\varepsilon}_2$ are all two-dimensional vectors where $\boldsymbol{a}_1 = \begin{bmatrix} a_{11} & a_{12} \end{bmatrix}^{\text{T}}$ and $\boldsymbol{a}_2 = \begin{bmatrix} a_{21} & a_{22} \end{bmatrix}^{\text{T}}$. The vector relations above can be repeated in their scalar components

$$\begin{aligned} \begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix} &= \begin{bmatrix} a_{\text{pcom}_1} \\ a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{12} \end{bmatrix} \\ \begin{bmatrix} a_{21} \\ a_{22} \end{bmatrix} &= \begin{bmatrix} a_{\text{pcom}_1} \\ a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} + \begin{bmatrix} \varepsilon_{21} \\ \varepsilon_{22} \end{bmatrix} \end{aligned} \tag{19}$$

or the first rows and the second rows may be combined separately as shown below:

$$\begin{aligned} \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} &= \begin{bmatrix} a_{\text{pcom}_1} \\ a_{\text{pcom}_1} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \end{bmatrix} \\ \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} &= \begin{bmatrix} a_{\text{pcom}_2} \\ a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} + \begin{bmatrix} \varepsilon_{12} \\ \varepsilon_{22} \end{bmatrix} \end{aligned} \tag{20}$$

Or with the first and second rows, our standard form of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}$ can be easily written

$$\begin{aligned} \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_1} \\ b_{11} \\ b_{21} \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \end{bmatrix} \\ \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_2} \\ b_{12} \\ b_{22} \end{bmatrix} + \begin{bmatrix} \varepsilon_{12} \\ \varepsilon_{22} \end{bmatrix} \end{aligned} \tag{21}$$

In conclusion, the first components in the math model can be combined to find the first component in $\boldsymbol{a}_{\text{pcom}}$. Similarly, the second components can be combined to find the second component in $\boldsymbol{a}_{\text{pcom}}$. Another fortunate event is that the matrix $\boldsymbol{A}$ is the same in both relations. The above example can be extended to any n-dimensional vectors with ease by writing similar equations with the first, second, etc., and finally nth components of all the vectors in the training set.

4.2 Derivation of the First Form of the Pseudo-Common Vector

In this section, we will try to derive a general equation for calculating the pseudo-common vectors using an induction method. This method will start with one of the simplest cases given in the example of the previous section where the (feature) vectors were all two dimensional. The matrix $\boldsymbol{A}$ was calculated as

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

If the feature vector $\boldsymbol{a}_x$ is one of the test vectors whose class is to be identified, then keeping in mind the example of the previous section, a component-wise simplified form of Eq. (13) can be written as

$$\begin{aligned} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{x1} \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a'_{\text{pcom}_1} \\ b'_{11} \\ b'_{21} \\ b'_{x1} \end{bmatrix} + \begin{bmatrix} \varepsilon'_{11} \\ \varepsilon'_{21} \\ \varepsilon_{x_1} \end{bmatrix} \\ \begin{bmatrix} a_{12} \\ a_{22} \\ a_{x2} \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a'_{\text{pcom}_2} \\ b'_{12} \\ b'_{22} \\ b'_{x2} \end{bmatrix} + \begin{bmatrix} \varepsilon'_{12} \\ \varepsilon'_{22} \\ \varepsilon_{x_2} \end{bmatrix}. \end{aligned} \tag{22}$$

If the training set has only two-dimensional two vectors for a class as above, then the matrix $\boldsymbol{A}'$ can be written as:

$$\boldsymbol{A}' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Another simplification is related to the pseudo-inverses of the matrices $\boldsymbol{A}$ and $\boldsymbol{A}'$. For example, if

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \text{then} \quad \boldsymbol{A}^{+} = \begin{bmatrix} 1/3 & 1/3 \\ 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix} \tag{23}$$

or if

$$\boldsymbol{A}=\begin{bmatrix}1&1&0&0\\1&0&1&0\\1&0&0&1\end{bmatrix},\quad \text{then}\quad \boldsymbol{A}^{+}=\begin{bmatrix}1/4&1/4&1/4\\3/4&-1/4&-1/4\\-1/4&3/4&-1/4\\-1/4&-1/4&3/4\end{bmatrix} \tag{24}$$

The matrices $\boldsymbol{A}$ and their pseudo-inverses have standard forms for any dimensionality. If m is the number of vectors in the training set from which we want to calculate $\boldsymbol{a}_{\text{pcom}}$, then each component of the pseudo-common vector can be calculated from the following relation:

$$\begin{bmatrix}a_{\text{pcom}_i}\\b_{1i}\\b_{2i}\\\vdots\\b_{mi}\end{bmatrix}=\boldsymbol{A}^{+}\begin{bmatrix}a_{1i}\\a_{2i}\\\vdots\\a_{mi}\end{bmatrix}\quad \text{for}\quad i=1,2,\ldots,n. \tag{25}$$

Therefore, a_{pcom_i} is just a scalar multiplication of the first row of $\boldsymbol{A}^{+}$ with the vector composed from the ith components of the feature vectors. That is, when $m=2$, $a_{\text{pcom}_i}=\begin{bmatrix}\frac{1}{3}&\frac{1}{3}\end{bmatrix}\begin{bmatrix}a_{1i}\\a_{2i}\end{bmatrix}$ for $i=1,2,\ldots,n$.

Or when $m=3$, $a_{\text{pcom}_i}=\begin{bmatrix}\frac{1}{4}&\frac{1}{4}&\frac{1}{4}\end{bmatrix}\begin{bmatrix}a_{1i}\\a_{2i}\\a_{3i}\end{bmatrix}$ for $i=1,2,\ldots,n$.

Or, in general, by induction for any value of m, where m is the number of vectors in the training set,

$$a_{\text{pcom}_i}=\begin{bmatrix}\frac{1}{m+1}&\frac{1}{m+1}&\cdots&\frac{1}{m+1}\end{bmatrix}\begin{bmatrix}a_{1i}\\a_{2i}\\\vdots\\a_{mi}\end{bmatrix}\quad \text{for}\quad i=1,2,\ldots,n. \tag{26}$$

This final relation saves us from the calculation burden of the pseudo-inverses of $\boldsymbol{A}$. Thus, it greatly reduces the number of operations.

One must be careful in the calculation of $\boldsymbol{a}'_{\text{pcom}}$ whenever $\boldsymbol{a}_x$ is added. That is,

$$a'_{\text{pcom}_i}=\begin{bmatrix}\frac{1}{m+2}&\frac{1}{m+2}&\cdots&\frac{1}{m+2}&\frac{1}{m+2}\end{bmatrix}\begin{bmatrix}a_{1i}\\a_{2i}\\\vdots\\a_{mi}\\a_{xi}\end{bmatrix}\quad \text{for}\quad i=1,2,\ldots,n. \tag{27}$$

This result is due to the change of $\boldsymbol{A}$ to $\boldsymbol{A}'$. Notice that the relations given in Eqs. (26) and (27) are close to the average value relations given in the first part of "Appendix." The general form of the pseudo-inverse of the matrix $\boldsymbol{A}$ is derived in the second part of "Appendix."

4.3 Distance in the Decision Criterion

The component-wise relations given in the previous subsection can be written in vector form. If there are m feature vectors in a class C, that is, $C=\{\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_m\}$, then

$$\boldsymbol{a}_{\text{pcom}}=\frac{1}{m+1}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m\right) \tag{28}$$

or when we want to calculate $\boldsymbol{a}'_{\text{pcom}}$ with the addition of $\boldsymbol{a}_x$

$$\boldsymbol{a}'_{pcom}=\frac{1}{m+2}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m+\boldsymbol{a}_x\right). \tag{29}$$

After being able to calculate $\boldsymbol{a}_{\text{pcom}}$ and $\boldsymbol{a}'_{\text{pcom}}$ from their simplified formulas given in Eqs. (28) and (29), one may start to wonder when the distance of any test vector $\boldsymbol{a}_x$ to the class C is equal to zero. A zero distance to a class C will assure us that $\boldsymbol{a}_x$ belongs to that class. Then, the following theorem will be useful and can be proved with ease.

Theorem *If* $\boldsymbol{a}_x=\boldsymbol{a}_{\text{pcom}}$, *then* $\boldsymbol{a}'_{\text{pcom}}=\boldsymbol{a}_{\text{pcom}}$.

Proof Starting with Eq. (29)

$$\boldsymbol{a}'_{\text{pcom}}=\frac{1}{m+2}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m+\boldsymbol{a}_x\right) \tag{30}$$

$$\boldsymbol{a}_x=\boldsymbol{a}_{\text{pcom}}\quad \text{means that}\quad \boldsymbol{a}_x=\frac{1}{m+1}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m\right)$$

as it is given in Eq. (28).

By substituting this final relation in the previous one,

$$\boldsymbol{a}'_{\text{pcom}}=\frac{1}{m+2}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m+\frac{1}{m+1}\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m\right)\right). \tag{31}$$

Multiply and divide the first m terms in the above summation with $(m+1)$ to get

$$\boldsymbol{a}'_{\text{pcom}}=\frac{\left((m+1)\boldsymbol{a}_1+(m+1)\boldsymbol{a}_2+\cdots+(m+1)\boldsymbol{a}_m+\left(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_m\right)\right)}{(m+1)(m+2)} \tag{32}$$

$(m+2)$ factors will be canceled in the numerator and the denominator to get

$$\boldsymbol{a}'_{\text{pcom}} = \frac{1}{m+1}\left(\boldsymbol{a}_1 + \boldsymbol{a}_2 + \cdots + \boldsymbol{a}_m\right) = \boldsymbol{a}_{\text{pcom}} = \boldsymbol{a}_x \tag{33}$$

□

This theorem tells us that the distance of $\boldsymbol{a}_x$ to the class C will be zero whenever $\boldsymbol{a}_x = \boldsymbol{a}_{\text{pcom}}$ due to the distance formula given in Eq. (16), $d_x^2 = \left\|\boldsymbol{a}'_{\text{pcom}} - \boldsymbol{a}_{\text{pcom}}\right\|^2$.

5 Computation of the Second Form of the Pseudo-Common Vector

In the previous sections, we demonstrated what we called the first form of the pseudo-common vector. However, a second form of the pseudo-common vector can also be calculated using pseudo-inverses and by playing with the mathematical model that was given in Eq. (10). Our first attempt is to imbed the error vectors into the difference vectors in Eq. (10) as in [16]; then, we will get the following:

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{a}_{\text{pcom}} + \boldsymbol{b}_1^{\text{new}} \\ \boldsymbol{a}_2 &= \boldsymbol{a}_{\text{pcom}} + \boldsymbol{b}_2^{\text{new}} \\ &\vdots \\ \boldsymbol{a}_m &= \boldsymbol{a}_{\text{pcom}} + \boldsymbol{b}_m^{\text{new}} \end{aligned} \tag{34}$$

where $\boldsymbol{b}_i^{\text{new}} = \boldsymbol{b}_i + \boldsymbol{\varepsilon}_i$ for $i = 1, 2, \ldots, m$. The relation above can also be written in the vector–matrix form as:

$$\begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_m \\ \boldsymbol{a}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} \\ \vdots \\ \boldsymbol{I}_{n\times n} \end{bmatrix} \left[\boldsymbol{a}_{\text{pcom}}\right] + \begin{bmatrix} \boldsymbol{b}_1^{\text{new}} \\ \boldsymbol{b}_2^{\text{new}} \\ \vdots \\ \boldsymbol{b}_m^{\text{new}} \end{bmatrix}. \tag{35}$$

As previously stated, the vector–matrix relation is not useful in our computations due to its high dimensionality. The component-wise relations will be very useful for our calculations as demonstrated in the example of Sect. 4.1 in Eq. (20). For the training set with two features, we will get

$$\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} = \begin{bmatrix} a_{\text{pcom}_1} \\ a_{\text{pcom}_1} \end{bmatrix} + \begin{bmatrix} b_{11}^{\text{new}} \\ b_{21}^{\text{new}} \end{bmatrix} \tag{36}$$

for the first components. For the second components, we will have the following:

$$\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \begin{bmatrix} a_{\text{pcom}_2} \\ a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} b_{12}^{\text{new}} \\ b_{22}^{\text{new}} \end{bmatrix}. \tag{37}$$

A more general relation for this case when the number of feature vectors is equal to m will be

$$\begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{bmatrix} = \left[\boldsymbol{1}_{m\times 1}\right] a_{\text{pcom}_i} + \begin{bmatrix} b_{1i}^{new} \\ b_{2i}^{new} \\ \vdots \\ b_{mi}^{\text{new}} \end{bmatrix} \quad \text{for} \quad i = 1, 2, \ldots, n \tag{38}$$

where the index i indicates the number of each component. Then, the minimization of the sum of the norm squares of the difference vectors with respect to the pseudo-common vector can be achieved using pseudo-inverses, that is,

$$a_{\text{pcom}_i} = \left(\left[\boldsymbol{1}_{m\times 1}\right]^+\right) \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{bmatrix} \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{39}$$

From here, we can deduce that each component of the $\boldsymbol{a}_{\text{pcom}}$ can be calculated as the average value of the related components of the feature vectors in the training set due to the following reason:

$$\left[\boldsymbol{1}_{m\times 1}\right]^+ = \frac{1}{m}\left[\boldsymbol{1}_{1\times m}^{\text{T}}\right] \text{ and } a_{\text{pcom}_i} = \frac{1}{m}\left[\boldsymbol{1}_{1\times m}^{\text{T}}\right] \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{bmatrix} \tag{40}$$

for $\quad i = 1, 2, \ldots, n.$

Or, in general, the second form of the pseudo-common vector will be the following:

$$\boldsymbol{a}_{\text{pcom}} = \frac{1}{m}\left(\boldsymbol{a}_1 + \boldsymbol{a}_2 + \cdots + \boldsymbol{a}_m\right) = \boldsymbol{a}_{\text{ave}} \tag{41}$$

One can notice the similarity and the dissimilarity between the two forms of the pseudo-common vectors in Eqs. (28) and (41) with ease. This is mentioned in the first part of "Appendix." The distance of $\boldsymbol{a}_x$ to a class with an average value of $\boldsymbol{a}_{\text{ave}}$ is calculated from the following relation:

$$d_x^2 = \left\|\boldsymbol{a}_x - \boldsymbol{a}_{\text{ave}}\right\|^2 \tag{42}$$

The distance in Eq. (42) is calculated for each class, and the test vector $\boldsymbol{a}_x$ is classified according to the criterion below:

$$C_x = \underset{1\le j\le C}{\text{argmin}} \left\|\boldsymbol{a}_x - \boldsymbol{a}_{\text{ave}}^j\right\|^2 \tag{43}$$

where j denotes the class number. That is, the vector $\boldsymbol{a}_x$ will be assigned to the class where it has a minimum distance.

6 Experimental Work on Three Databases for Classification

Three databases, TI-Digit, AR-Face, and MNIST, were used in order to evaluate the performance of the proposed method. Explanations about three databases and the experimental results are given in the following subsections.

6.1 Experimental Work with the TI-Digit Database

The TI-Digit database includes 11 isolated digits. Our TI-training and TI-test sets have 112 and 111 speakers, respectively. Each speaker in these sets repeats each digit twice [31]. The training set used in this study contains 224 repetitions from the TI-training set and 202 from the TI-test set for each digit. The test set contains the remaining 20 repetitions in the TI-test set for each digit. Therefore, the repetitions or the feature vectors in the test and training sets are completely separate in the experiments. An identical number of male and female speakers are used in the training set (213 male and 213 female) and the test set (10 male and 10 female).

After end-point detection, the isolated speech is divided into a frame of 256 samples with one-fourth overlap and the frames are pre-emphasized. Furthermore, eleven root-melcep parameters are extracted from each frame [32]. The mel-frequency cepstrum (MFC) parameters are the coefficients of a speech signal's short-term power spectrum, relying on the transformation of a log power spectrum on a nonlinear frequency called the mel scale. Finally, roots of the resulting parameters are taken. The feature vector for each repetition is obtained from the stack of the frame parameters. Since the dimension of the feature vectors differs for every digit, the length normalization is obtained by padding random values to the end of the feature vectors if the dimension of the feature vector is less than 407×1. The short-duration utterances such as the digits "four," "three," and "ow" need more padding. The average number of padded samples for these digits is 242.15, 239.74, and 229.28, respectively. The overall percentage of padding in the database is 50.7%. Since 20 feature vectors in the test set are too limited to evaluate the recognition performance, the leave-twenty-out strategy is used. Thus, the testing stage is repeated 11 times to consider all the repetitions in the TI-test set. The average recognition rates of all digits are given in Table 1 for each fold.

Table 1 The recognition rates (%) attained for the TI-Digit database

"Leave-Twenty-Out" Fold number	Average vector		The first form of the pseudo-common vector	
	Training	Test	Training	Test
1	90.33	84.09	90.33	84.09
2	89.97	91.82	89.97	91.82
3	90.12	90.45	90.10	90.45
4	89.99	92.27	89.97	92.27
5	90.20	85.91	90.18	85.91
6	89.99	92.27	90.03	92.27
7	90.14	88.18	90.14	88.19
8	90.14	91.82	90.14	91.82
9	89.93	94.55	89.95	95.00
10	89.88	92.73	89.88	92.73
11	90.06	89.55	90.06	89.55
Average	90.07	90.33	90.07	90.37

6.2 Experimental Work with the AR-Face Database

The AR-Face database was constructed by two scientists, Martinez and Benavente [33]. This collection comprises of the RGB (red–green–blue) formatted face images of 126 distinct persons. The face poses include distinct expressions of faces, lighting situations, or irregular occlusions. There are 26 different facial poses, thirteen of which are shot in the first session and the remaining in the second session for each person. The image poses are saved in the matrices with the size of 576×768 pixels. In the experimental work, first, the RGB-formatted poses were transformed into black-white poses with 8-bit representation. Their sizes are reduced into 143×191 pixels as illustrated in Fig. 1. All the faces are aligned by localizing the eyes and noses of persons. Then, a cropping operation is performed on all resized poses. They are cropped to the dimension of 100×85. Figure 2 includes only face portions of arbitrarily selected five persons. Accordingly, 30 male and 20 female persons are arbitrarily selected and their 14 poses containing distinctive facial expressions and lighting situations are used in the experiments. In total, 700 face poses are used. A cross-validation procedure is applied for the classification where 13 out of 14 facial images of every person are separated for the training, and the other one facial image is separated for the testing in each step of the cross-validation. Consequently, the cross-validation procedure comprises 14 different steps. The average classification rates of all persons in each cross-validation step are presented in Table 2.

Subsequently, the poses affected by the maximum illumination (two-side illumination) are removed so that one pose per session is not involved in the new classification scheme. Therefore, a cross-validation procedure is applied for the classification where 11 out of 12 facial images of every person are separated for the training, and the remaining one facial image is separated for the testing steps. The average classification results of all persons in each cross-validation step are given in Table 3.

6.3 Experimental Work with the MNIST Database

MNIST is a handwritten digit database with 70,000 samples, of which 60,000 are training and 10,000 are test samples. It is a subset of a larger handwritten character database called NIST [34]. The digit classes of NIST are used to generate MNIST [35]. These images are normalized and centered in an image with 28×28 pixels. The center of the gravity of intensity is fixed to the center of the image. Moreover, sample images from the MNIST database are given in Fig. 3. We

Fig. 1 The original images of five persons selected arbitrarily from the AR face database. (Each image has a size of 143×191)

executed experiments to investigate the speckle noise effect. In Fig. 4, sample images from the MNIST database and their speckle noise added versions for variances $\sigma = 0.01$ and $\sigma = 0.1$ are given. The experimental results for the MNIST database are given in Table 4. The average vector and the pseudo-common vector give the same results as expected since the number of the training vectors increases causing a convergence of the pseudo-common vector to the average vector.

7 Discussion

If the recognition accuracies given in Table 1 are analyzed, the results for the test vectors are slightly higher than that of training vectors for the TI-Digit database; this outcome is pleasing. These results show us that the first form of the pseudo-common vector approach is as good as the approach using the second form, that is, the average vectors, but it does not increase the recognition accuracies to the level of the other state-of-the-art recognition methods.

On the other hand, the training recognition rates are higher than those of test vectors for the AR-Face database (Table 2) as expected, because the number of training vectors is quite low. If one increases the size of the training vector, the accuracy of the test vectors may be close to that of training vectors and may reach to more satisfactory rates. Since the approach proposed in this paper is based on the least square estimation, more training vectors implicitly mean an influential minimization of error; therefore, we may conclude that the approach inherently needs a larger number (greater than the number of training vectors) of vectors in a training set because of its natural mathematical model. In addition, if the two poses, which include two-sided illumination effects—that is, the hardest illumination condition (an illumination source gives light to both sides of faces that can be seen on the seventh pose of each individual in either Fig. 1 or 2)—are removed from the database, the total number of face poses decreases to 12 for each person. Subsequently, the average recognition rates are increased to 78.42 and 72.83 for the training and test sets, respectively, if all of the pseudo-common vector experiments are repeated for the AR face database where two poses including two-sided illumination effects are removed (Table 3). One can easily infer from these results that illuminated poses contain a large value of variance from the ensemble mean. Therefore, they are not compatible with the inherent nature of poses for each individual. It can also be concluded that the pseudo-common vector is more robust than the average vector under illumination conditions since the test set results of the pseudo-common vector are higher than those of the average vector for 14 poses.

The recognition rates in the MNIST database given in Table 4 are also lower compared with the state-of-the-art

Fig. 2 The face portions cropped from the original images given in Fig. 1. (Each image has a size of 100×85)

Table 2 The recognition rates (%) attained for the AR face database

"Leave-One-Out" Fold number	Average vector		First form of the pseudo-common vector	
	Training	Test	Training	Test
1	70.31	90	70.15	94
2	71.38	84	69.85	94
3	72.31	76	70.15	94
4	72.92	58	71.54	74
5	71.54	84	72.62	52
6	74.15	54	74.31	42
7	78.00	8	75.54	6
8	71.54	76	69.69	98
9	71.38	66	70.31	90
10	71.85	70	70.92	88
11	74.00	42	72.46	66
12	71.23	90	71.23	72
13	72.31	68	73.54	46
14	77.08	16	75.54	10
Average	72.86	63	71.99	66.14

Table 3 The recognition rates (%) attained for the AR face database for 12 poses

"Leave-One-Out" Fold number	Average vector		The first form of the pseudo-common vector	
	Training	Test	Training	Test
1	81.10	94	76.55	96
2	81.10	94	77.09	94
3	81.45	92	77.09	98
4	82.90	70	77.45	84
5	83.81	52	81.45	30
6	85.63	38	81.27	24
7	80.73	98	77.09	100
8	81.45	90	77.09	98
9	82.55	88	77.45	96
10	83.27	66	77.27	88
11	81.99	74	80.36	38
12	86.00	42	80.91	28
Average	82.67	74.83	78.42	72.83

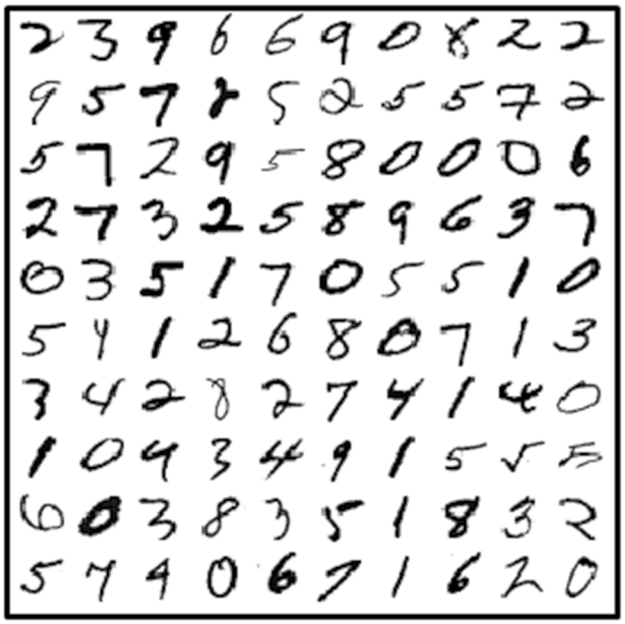

Fig. 3 Sample images of the MNIST database

values, but they are not as bad. In fact, one may deduce that adding a little noise ($\sigma = 0.01$) to the numerical characters helps us distinguish the numbers better compared with the clean numerals, whereas adding a higher level of noise ($\sigma = 0.1$) starts to degrade recognition rates. Some of the reported classification performances of the state-of-the-art methods in the three databases are given in Table 5. These

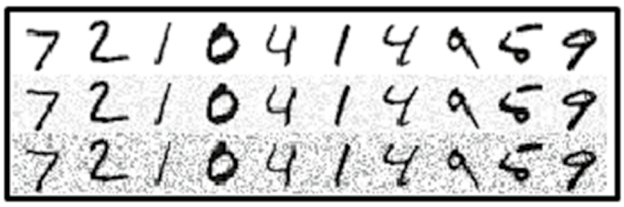

Fig. 4 Sample images and their speckle noise added variations. The first row for clean images, the second row for noisy images with $\sigma = 0.01$, and the third row for noisy images with $\sigma = 0.1$

Table 4 The recognition rates (%) attained for the MNIST database

	Average vector	Pseudo-common vector
Training	80.8	80.8
Clean test	82.04	82.04
Noisy test ($\sigma = 0.01$)	82.41	82.41
Noisy test ($\sigma = 0.1$)	80.52	80.52

results show that the pseudo-common vector approach does not increase the recognition accuracies to the level of the other state-of-the-art recognition methods. The results can only be compared with methods that use the average values of the classes. However, the main reason why the methodology in the paper is introduced is to show the possibility of using a completely innovative approach in the pattern recognition problems.

8 Conclusion

In this paper, we proposed the pseudo-common vector approach in the context of the pattern recognition area. The pseudo-common vector is defined as a nonzero and unique vector for a class. The previous implementation of common vectors comprised the same properties. However, in the previous work, the calculation of common vectors was based on long-continued eigenvalue–eigenvector decompositions. Therefore, the process was realized in a complex manner although the modeling of feature vectors is quite simple.

This paper introduces a pseudo-inverse-based approach to find common vectors which represent a class appropriately. In mathematics, and particularly in linear algebra, a pseudo-inverse is a generalization of the inverse of a matrix. A popular use of pseudo-inverse is to compute a best fit (least squares) solution to a system of linear equations that lacks a unique solution, or, in other words to find a minimum norm solution to a system of linear equations with multiple solutions. This paper introduces substantially simple ways to compute a pseudo-common vector of a class using the pseudo-inverse concept. The application of this concept led us to Eqs. (28) and (29). The first form of the pseudo-common vectors is obtained by dividing the sum of all training vectors in a class into their total number plus one ($m+1$). Thus, it is very close to the class averages, which are the second form of the pseudo-common vectors, especially when the number of feature vectors in the training set is large. In other words, only an ordinary division operation is required in order to calculate the first and the second forms of the pseudo-common vectors

Table 5 State-of-the-art classification results (%) on the TI-Digit, AR, and MNIST databases

TI-Digit		AR		MNIST	
HTK [36]	99.11	Eigenface [18]	79.14	LeNet-4 [37]	98.90
HMM [38]	98.24	Fisherface [18]	98.85	LeNet-5 [37]	99.05
CMN [39]	98.56	Direct-LDA [18]	98.64	Virtual SV [37]	99.00
HTK [40]	99.53	DCVA [18]	99.35	Pair-wise SVC [41]	98.48
CD-HMM [42]	99.80	2DPCA [43]	89.80	Dong et al. [44]	99.01
		SRC [45]	94.70	Belongie et al. [46]	99.37
		SVM [47]	97.60	Teow et al. [48]	99.41
		CRC_LRS [49]	93.70	Liu et al. [50]	99.18
		LC-KSVD2 [51]	97.80	Mayraz et al. [52]	98.30

of a class. Hence, a practical and computationally easy approach is proposed.

In future work, we will try to show how improvements can be made in pseudo-common vector approaches in order to increase the recognition rates at least to the high values obtained in our previous work [4].

Appendix A

Average Values

The relations given in Eq. (20) will be repeated below for convenience

$$\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} = \begin{bmatrix} a_{\text{pcom}_1} \\ a_{\text{pcom}_1} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \end{bmatrix}$$

$$\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \begin{bmatrix} a_{\text{pcom}_2} \\ a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} + \begin{bmatrix} \varepsilon_{12} \\ \varepsilon_{22} \end{bmatrix}.$$

The above relations may be written as:

$$\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \end{bmatrix}$$

$$\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} + \begin{bmatrix} \varepsilon_{12} \\ \varepsilon_{22} \end{bmatrix}.$$

The differences above can be combined with the errors:

$$\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{11} + \varepsilon_{11} \\ b_{21} + \varepsilon_{21} \end{bmatrix}$$

$$\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} a_{\text{pcom}_2} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{12} + \varepsilon_{12} \\ b_{22} + \varepsilon_{22} \end{bmatrix}$$

Assuming that all the errors are imbedded in the difference components of the feature vectors, then the minimization of the sum of the norm squares of the difference vectors with respect to the pseudo-common vectors can be achieved by using pseudo-inverses, that is,

$$a_{\text{pcom}_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^{+} \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} \quad \text{and} \quad a_{\text{pcom}_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^{+} \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}$$

or else

$$a_{\text{pcom}_1} = \begin{bmatrix} 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} \quad \text{and} \quad a_{\text{pcom}_2} = \begin{bmatrix} 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}.$$

In the above example, we may conclude as the first component of the common vector is equal to the average value of the first components of the feature vectors in the training set. And the same thing is true for the second components. We should just remember that multiplying both sides with the pseudo-inverse of the matrix of $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ minimizes the norm squares of the common vectors. Or in general,

$$\boldsymbol{a}_{\text{pcom}} = \frac{1}{m}\left(\boldsymbol{a}_1 + \boldsymbol{a}_2 + \cdots + \boldsymbol{a}_m\right) = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{a}_j = \boldsymbol{a}_{\text{ave}}$$

minimizes the norm squares of the pseudo-common vectors. This was given with the **Criterion 1** in our earliest paper [1]. In our new approach, the first form of the pseudo-common vector is a bit different than the average vector. The divisor in the new approach is equal to $m+1$ instead of m. If m is a small number, then the pseudo-common vector will be far different than the average vector. But if m is a large number, then the pseudo-common vector and the average vector will get close to each other.

Appendix B

A General Solution of the Pseudo-Inverse of the Matrix *A*

The matrix $\boldsymbol{A}$ was given as $\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ in Eq. (23) and $\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$ in Eq. (24) when the number of data was equal to $m = 2$ and $m = 3$, respectively. Whenever we want to write the matrix $\boldsymbol{A}$ for any value of m, then we will obtain the general form of $\boldsymbol{A}$ as follows:

$\boldsymbol{A} = \begin{bmatrix} \mathbf{1}_{m\times 1} & \boldsymbol{I}_{m\times m} \end{bmatrix} \in R^{m\times(m+1)}$ where $\mathbf{1}_{m\times 1}$ is a vector whose components are all ones with the dimensions given in its own subscript and $\boldsymbol{I}_{m\times m}$ is an identity matrix whose dimensions are also given in its own subscript. We may decompose the matrix $\boldsymbol{A}$ into two matrices

$\boldsymbol{B} = \begin{bmatrix} \mathbf{1}_{m\times 1} & \mathbf{0}_{m\times m} \end{bmatrix} \in R^{m\times(m+1)}$ and $\boldsymbol{C} = \begin{bmatrix} \mathbf{0}_{m\times 1} & \boldsymbol{I}_{m\times m} \end{bmatrix} \in R^{m\times(m+1)}$ so that $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}$.

We may also write their transposes with ease

$\boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} \mathbf{1}^{\mathrm{T}}_{1\times m} \\ \boldsymbol{I}_{m\times m} \end{bmatrix} \in R^{(m+1)\times m}$, $\boldsymbol{B}^{\mathrm{T}} = \begin{bmatrix} \mathbf{1}^{\mathrm{T}}_{1\times m} \\ \mathbf{0}_{m\times m} \end{bmatrix} \in R^{(m+1)\times m}$, $\boldsymbol{C}^{\mathrm{T}} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}}_{1\times m} \\ \boldsymbol{I}_{m\times m} \end{bmatrix} \in R^{(m+1)\times m}$ so that $\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{B}^{\mathrm{T}} + \boldsymbol{C}^{\mathrm{T}}$.

The multiplication of $\boldsymbol{A}$ with its transpose will give us a square matrix of $m \times m$

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}.$$

Since $\boldsymbol{B}\boldsymbol{C}^{\mathrm{T}}$ and $\boldsymbol{C}\boldsymbol{B}^{\mathrm{T}}$ both will give zero matrices, the above relation may be rewritten as:

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{C}\boldsymbol{C}^{\mathrm{T}} = \mathbf{1}_{m\times 1}\mathbf{1}_{1\times m}^{\mathrm{T}} + \boldsymbol{I}_{m\times m}\boldsymbol{I}_{m\times m}.$$

Also $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \mathbf{1}_{1\times m}^{\mathrm{T}}\mathbf{1}_{m\times 1} = m$ is the total number of data used in the matrix $\boldsymbol{A}$. The only nonzero eigenvalue of $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}$ is equal to m, and all its other eigenvalues are zeros since $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}$ has rank 1.

The singular value decomposition of the matrix $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{U}_A\boldsymbol{S}_A\boldsymbol{V}_A^{\mathrm{T}}$. Then,

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{U}_A\boldsymbol{S}_A\boldsymbol{V}_A^{\mathrm{T}}\boldsymbol{V}_A\boldsymbol{S}_A^{\mathrm{T}}\boldsymbol{U}_A^{\mathrm{T}}$$

where $\boldsymbol{V}_A^{\mathrm{T}}\boldsymbol{V}_A = \boldsymbol{I}_{(m+1)\times(m+1)}$ and $\boldsymbol{S}_A\boldsymbol{S}_A^{\mathrm{T}} = \boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^T}$ are diagonal matrices. The eigenvalues of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ will be the diagonal elements in $\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}}$. Then, $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ can be written as:

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{U}_A\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^T}\boldsymbol{U}_A^{\mathrm{T}} = \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{I}_{m\times m} \tag{B.1}$$

Premultiplying all sides of Eq. (B.1) with $\boldsymbol{U}_A^{\mathrm{T}}$ and postmultiplying with $\boldsymbol{U}_A$ will give us the following:

$$\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}} = \boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{U}_A + \boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{U}_A = \boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{U}_A + \boldsymbol{I}_{m\times m}.$$

Therefore, $\boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{U}_A$ is a diagonal matrix with the diagonal elements of m and all zeros, that is,

$$\boldsymbol{\Lambda}_{\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}} = \boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{U}_A = \begin{bmatrix} m & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \\ \mathbf{0}_{(m-1)\times 1} & \mathbf{0}_{(m-1)\times(m-1)} \end{bmatrix}.$$

Then,

$$\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}} = \boldsymbol{\Lambda}_{\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}} + \boldsymbol{I}_{m\times m} = \begin{bmatrix} m+1 & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} \end{bmatrix}.$$

After determining all the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$, we may write $\boldsymbol{S}_A$ in terms of singular values as follows:

$$\boldsymbol{S}_A = \begin{bmatrix} \sqrt{m+1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} & 0_{1\times 1} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} & \mathbf{0}_{(m-1)\times 1} \end{bmatrix}.$$

Then, the pseudo-inverse of $\boldsymbol{S}_A$ can be written as:

$$\boldsymbol{S}_A^{+} = \begin{bmatrix} 1/\sqrt{m+1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} \\ 0_{1\times 1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \end{bmatrix}.$$

Now we have two relations to calculate the pseudo-inverse of $\boldsymbol{A}$ by using SVD,

1. $\boldsymbol{A} = \boldsymbol{U}_A\boldsymbol{S}_A\boldsymbol{V}_A^{\mathrm{T}}$
2. $\boldsymbol{A}^{+} = \boldsymbol{V}_A\boldsymbol{S}_A^{+}\boldsymbol{U}_A^{\mathrm{T}}$

If we represent $\boldsymbol{U}_A$ and $\boldsymbol{V}_A$ with their singular value vectors, $\boldsymbol{A}$ and $\boldsymbol{A}^{+}$ can be written as follows:

1. $$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_m \end{bmatrix} \begin{bmatrix} \sqrt{m+1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} & 0_{1\times 1} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} & \mathbf{0}_{(m-1)\times 1} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_{m+1}^{\mathrm{T}} \end{bmatrix}$$
$$= \sqrt{m+1}\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}} + \cdots + \boldsymbol{u}_m\boldsymbol{v}_m^{\mathrm{T}} \tag{B.2}$$

2. $$\boldsymbol{A}^{+} = \boldsymbol{V}_A \begin{bmatrix} 1/\sqrt{m+1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} \\ 0_{1\times 1} & \mathbf{0}_{1\times(m-1)}^{\mathrm{T}} \end{bmatrix} \boldsymbol{U}_A^{\mathrm{T}}$$
$$= \frac{1}{\sqrt{m+1}}\boldsymbol{v}_1\boldsymbol{u}_1^{\mathrm{T}} + \boldsymbol{v}_2\boldsymbol{u}_2^{\mathrm{T}} + \cdots + \boldsymbol{v}_m\boldsymbol{u}_m^{\mathrm{T}} \tag{B.3}$$

From Eq. (B.2), the transpose of $\boldsymbol{A}$ can be found as:

$$\boldsymbol{A}^{\mathrm{T}} = \sqrt{m+1}\boldsymbol{v}_1\boldsymbol{u}_1^{\mathrm{T}} + \boldsymbol{v}_2\boldsymbol{u}_2^{\mathrm{T}} + \cdots + \boldsymbol{v}_m\boldsymbol{u}_m^{\mathrm{T}}.$$

The last $(m-1)$ outer product terms are the same in Eqs. (B.2) and (B.3). Let us define them with the matrix $\boldsymbol{E}$ and the matrix $\boldsymbol{D}$ as:

$$\boldsymbol{E} = \boldsymbol{v}_2\boldsymbol{u}_2^{\mathrm{T}} + \cdots + \boldsymbol{v}_m\boldsymbol{u}_m^{\mathrm{T}}$$

$$\boldsymbol{D} = \boldsymbol{v}_1\boldsymbol{u}_1^{\mathrm{T}}.$$

Then, Eqs. (B.2) and (B.3) can be written in terms of $\boldsymbol{D}$ and $\boldsymbol{E}$ as follows:

1. $$\boldsymbol{A}^{\mathrm{T}} = \sqrt{m+1}\boldsymbol{D} + \boldsymbol{E}. \tag{B.4}$$

2. $$\boldsymbol{A}^{+} = \frac{1}{\sqrt{m+1}}\boldsymbol{D} + \boldsymbol{E}. \tag{B.5}$$

The following relation is reached by subtracting Eqs. (B.4) and (B.5) side by side:

$$\boldsymbol{A}^{+} = \boldsymbol{A}^{\mathrm{T}} - \frac{m}{\sqrt{m+1}}\boldsymbol{D} = \boldsymbol{A}^{\mathrm{T}} - \frac{m}{\sqrt{m+1}}\boldsymbol{v}_1\boldsymbol{u}_1^{\mathrm{T}} \tag{B.6}$$

This final relation tells us that two singular value vectors $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ must be calculated to obtain all the elements of $\boldsymbol{A}^{+}$.

Calculation of $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$

We will start with

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{U}_A\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^T}\boldsymbol{U}_A^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_m \end{bmatrix}\boldsymbol{\Lambda}_{\boldsymbol{A}\boldsymbol{A}^T}\begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_m^{\mathrm{T}} \end{bmatrix}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{AA}^{\mathrm{T}}} = \begin{bmatrix} m+1 & \mathbf{0}^{\mathrm{T}}_{1\times(m-1)} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} \end{bmatrix}$.

In terms of outer products, $\boldsymbol{AA}^{\mathrm{T}} = (m+1)\boldsymbol{u}_1\boldsymbol{u}_1^{\mathrm{T}} + \boldsymbol{u}_2\boldsymbol{u}_2^{\mathrm{T}} + \cdots + \boldsymbol{u}_m\boldsymbol{u}_m^{\mathrm{T}}$. Postmultiplying both sides by $\boldsymbol{u}_1$, we will get the following:

$$\boldsymbol{AA}^{\mathrm{T}}\boldsymbol{u}_1 = (m+1)\boldsymbol{u}_1$$

which is an eigenvector problem. Using the constraint of $\|\boldsymbol{u}_1\| = 1$, we will get

$$\boldsymbol{u}_1^{\mathrm{T}} = \begin{bmatrix} \frac{1}{\sqrt{m}} & \frac{1}{\sqrt{m}} & \cdots & \frac{1}{\sqrt{m}} \end{bmatrix}.$$

In order to calculate $\boldsymbol{v}_1$, one needs to calculate $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ as $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{V}_A\boldsymbol{S}_A^{\mathrm{T}}\boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{U}_A\boldsymbol{S}_A\boldsymbol{V}_A^{\mathrm{T}}$. Since $\boldsymbol{U}_A^{\mathrm{T}}\boldsymbol{U}_A = \boldsymbol{I}$ and $\boldsymbol{S}_A^{\mathrm{T}}\boldsymbol{S}_A = \boldsymbol{\Lambda}_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ will be

$$\begin{aligned}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} &= \boldsymbol{V}_A\boldsymbol{\Lambda}_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}\boldsymbol{V}_A^{\mathrm{T}} \\ &= \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_{m+1} \end{bmatrix} \begin{bmatrix} m+1 & \mathbf{0}^{\mathrm{T}}_{1\times(m-1)} \\ \mathbf{0}_{(m-1)\times 1} & \boldsymbol{I}_{(m-1)\times(m-1)} \\ 0_{1\times 1} & \mathbf{0}^{\mathrm{T}}_{1\times m} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_{m+1}^{\mathrm{T}} \end{bmatrix}\end{aligned}$$

$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ can be written as the outer products of its eigenvectors as follow:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = (m+1)\boldsymbol{v}_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{v}_2\boldsymbol{v}_2^{\mathrm{T}} + \cdots + \boldsymbol{v}_m\boldsymbol{v}_m^{\mathrm{T}}.$$

By postmultiplying with $\boldsymbol{v}_1$, we will get the eigenvalue–eigenvector relation

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}_1 = (m+1)\boldsymbol{v}_1.$$

$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ has the form of

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} \mathbf{1}^{\mathrm{T}}_{1\times m}\mathbf{1}_{m\times 1} & \mathbf{1}^{\mathrm{T}}_{1\times(m-1)} \\ \mathbf{1}_{m\times 1} & \boldsymbol{I}_{m\times m} \end{bmatrix} = \begin{bmatrix} m & \mathbf{1}^{\mathrm{T}}_{1\times(m-1)} \\ \mathbf{1}_{m\times 1} & \boldsymbol{I}_{m\times m} \end{bmatrix}.$$

From the last two relations, $\boldsymbol{v}_1$ will be calculated as: $\boldsymbol{v}_1^{\mathrm{T}} = \begin{bmatrix} \frac{\sqrt{m}}{\sqrt{m+1}} & \frac{1}{\sqrt{m+1}} & \cdots & \frac{1}{\sqrt{m+1}} \end{bmatrix}$. Then,

$$\boldsymbol{D} = \boldsymbol{v}_1\boldsymbol{u}_1^{\mathrm{T}} = \begin{bmatrix} \frac{1}{\sqrt{m+1}} & \frac{1}{\sqrt{m+1}} & \cdots & \frac{1}{\sqrt{m+1}} \\ \frac{1}{m\sqrt{m+1}} & \frac{1}{m\sqrt{m+1}} & \cdots & \frac{1}{m\sqrt{m+1}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{m\sqrt{m+1}} & \frac{1}{m\sqrt{m+1}} & \cdots & \frac{1}{m\sqrt{m+1}} \end{bmatrix}.$$

Finally, $\boldsymbol{A}^{+}$ can be calculated by substituting $\boldsymbol{D}$ in the left-hand side of Eq. (B.6) as follows:

$$\begin{aligned}\boldsymbol{A}^{+} &= \begin{bmatrix} \mathbf{1}^{\mathrm{T}}_{1\times m} \\ \boldsymbol{I}_{m\times m} \end{bmatrix} - \begin{bmatrix} \frac{m}{m+1} & \frac{m}{m+1} & \cdots & \frac{m}{m+1} \\ \frac{1}{m+1} & \frac{1}{m+1} & \cdots & \frac{1}{m+1} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{m+1} & \frac{1}{m+1} & \cdots & \frac{1}{m+1} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{m+1} & \frac{1}{m+1} & \cdots & \frac{1}{m+1} \\ \frac{m}{m+1} & \frac{-1}{m+1} & \cdots & \frac{-1}{m+1} \\ \frac{-1}{m+1} & \frac{m}{m+1} & \cdots & \frac{-1}{m+1} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{-1}{m+1} & \frac{-1}{m+1} & \cdots & \frac{m}{m+1} \end{bmatrix} \in R^{(m+1)\times m}\end{aligned}$$

References

1. Samadi, F.; Akbarizadeh, G.; Kaabi, H.: Change detection in SAR images using deep belief network: a new training approach based on morphological images. IET Image Process. **13**(12), 2255–2264 (2019)
2. Norouzi, M.; Akbarizadeh, G.; Eftekhar, F.: A hybrid feature extraction method for SAR image registration. Signal Image Video Process. **12**(8), 1559–1566 (2018)
3. Moghaddam, A.; Akbarizadeh, G.; Kaabi, H.: Automatic detection and segmentation of blood vessels and pulmonary nodules based on a line tracking method and generalized linear regression model. Signal Image Video Process. **13**(3), 457–464 (2019)
4. Tirandaz, Z.; Akbarizadeh, G.: Unsupervised texture-based SAR image segmentation using spectral regression and Gabor filter bank. J. Indian Soc. Remote Sens. **44**(2), 177–186 (2016)
5. Andekah, Z.; Naderan, M.; Akbarizadeh, G.: Semi-supervised hyperspectral image classification using spatial-spectral features and superpixel-based sparse codes. In: Iranian Conference on Electrical Engineering (ICEE) (2017)
6. Akbarizadeh, G.; Rahmani, M.: Efficient combination of texture and color features in a new spectral clustering method for PolSAR image segmentation. Natl. Acad. Sci. Lett. **40**(2), 117–120 (2017)
7. Yuheng, S.; Hao, Y.: Image Segmentation Algorithms Overview. arXiv preprint arXiv:1707.02051 (2017)
8. Sharifzadeh, F.; Akbarizadeh, G.; Kavian, Y.: Ship classification in SAR images using a new hybrid CNN-MLP classifier. J. Indian Soc. Remote Sens. **47**(4), 551–562 (2019)
9. Akbarizadeh, G.: A new statistical-based kurtosis wavelet energy feature for texture recognition of SAR images. IEEE Trans. Geosci. Remote Sens. **50**(11), 4358–4368 (2012)
10. Akbarizadeh, G.; Tirandaz, Z.; Kooshesh, M.: A new curvelet-based texture classification approach for land cover recognition of SAR satellite images. Malays. J. Comput. Sci. **27**(3), 218–239 (2014)
11. Modava, M.; Akbarizadeh, G.; Soroosh, M.: Integration of spectral histogram and level set for coastline detection in SAR images. IEEE Trans. Aerosp. Electron. Syst. **55**(2), 810–819 (2019)
12. Akbarizadeh, G.; Modava, M.; Soroosh, M.: A novel hierarchical coastline detection in SAR images based on spectral-textural features and global-local information. IET Radar **13**(12), 2183–2195 (2019)

13. Ahmadi, N.; Akbarizadeh, G.: Iris tissue recognition based on GLDM feature extraction and hybrid MLPNN-ICA classifier. Neural Comput. Appl. **32**, 1–15 (2018)
14. Raeisi, A.; Akbarizadeh, G.; Mahmoudi, A.: Combined method of an efficient cuckoo search algorithm and nonnegative matrix factorization of different zernike moment features for discrimination between oil spills and lookalikes in SAR images. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. **11**(11), 4193–4205 (2018)
15. Taibi, F.; Akbarizadeh, G.; Farshidi, E.: Robust reservoir rock fracture recognition based on a new sparse feature learning and data training method. Multidimension. Syst. Signal Process. **30**, 2113–2146 (2019)
16. Gülmezoğlu, M.B.; Keskin, M.; Dzhafarov, V.; Barkana, A.: A novel approach to isolated word recognition. IEEE Trans. Speech Audio Process. **7**(6), 620–628 (1999)
17. Gülmezoğlu, M.B.; Dzhafarov, V.; Barkana, A.: The common vector approach and its relation to principal component analysis. IEEE Trans. Speech Audio Process. **9**(6), 655–662 (2001)
18. Cevikalp, H.; Neamtu, M.; Wilkes, M.; Barkana, A.: Discriminative common vector for face recognition. IEEE Trans. Pattern Anal. Mach. Intell. **27**(1), 4–13 (2005)
19. Lindfield, G.; Penny, J.: Linear equations and eigensystems. In: Numerical Methods using MATLAB, 3rd edn, pp. 67–145. Elsevier (2012)
20. Ford, W.: Numerical Linear Algebra with Applications Using MATLAB. Academic Press, New York (2014)
21. Li, L.; Gao, J.; Ge, H.: A new face recognition method via semi-discrete decomposition for one sample problem. Optik **127**(19), 7408–7417 (2016)
22. Katsikis, V.; Pappas, D.: Fast compuring of the Moore–Penrose inverse matrix. Electron. J. Linear Algebra **17**, 637–650 (2008)
23. Courrieu, P.: Fast computation of the Moore–Penrose inverse matrices. Neural Inf. Process. Lett. Rev. **8**(2), 25–29 (2005)
24. Schaik, A.; Tapson, J.: Online and adaptive pseudoinverse solutions for ELM weights. Neurocomputing **149**, 233–238 (2015)
25. Barata, J.; Hussein, M.: The Moore-Penrose pseudoinverse: a tutorial review of the theory. Braz. J. Phys. **42**(1–2), 146–165 (2012)
26. Gorecki, T.; Luczak, M.: Stacked regression with a generlization of the Moore–Penrose pseudoinverse. Stat. Transit. New Ser. **18**(3), 443–458 (2017)
27. Gorecki, T.; Luczak, M.: Evolutionarily tuned generalized pseudoinverse in linear discriminant analysis. Comput. Inform. **35**(3), 615–634 (2016)
28. Ricci, T.; Steiner, J.; Menezes, R.: NGC 7097: the active galactic nucleus and its mirror, revealed by principal component analysis tomography. Astrophys. J. Lett. **724**, 1–6 (2011)
29. Chountasis, S.; Katsikis, V.; Pappas, D.: Digital image reconstruction in the spectral domain utilizing the Moore–Penrose Inverse. Math. Prob. Eng. **724**, Article ID 750352 (2010)
30. Gallier, J., Quaitance, J.: Applications of SVD and pseudoinverses. Fundamentals of Linear Algebra and Optimization, pp. 443–465 (2018)
31. Gülmezoğlu, M.B.; Dzhafarov, V.; Edizkan, R.; Barkana, A.: The common vector approach and its comparison with other subspace methods in case of sufficient data. Comput. Speech Lang. **21**(2), 266–281 (2007)
32. Rabiner, L.; Schafer, R.: Digital Processing of Speech Signals. Prentice-Hall, Englewood Cliffs (1978)
33. Martinez, A.; Benavente, B.: The AR face database. CVC Technical Report #24 (1998)
34. Grother, P.: Nist special database 19—handprinted forms and characters database. National Institute of Standards and Thechnology (NIST) (1995)
35. Lecun, Y.; Cortes, C.: The MNIST database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/. Accessed 3 Jan 2020
36. Benjamin, J.; Kuldip, K.: Feature extraction from higher-lag autocorrelation coefficients for robust speech recognition. Speech Commun. **48**, 1458–1485 (2006)
37. Lecun, Y.; Jackel, L.; Bottou, L.; et al.: Comparison of learning algorithms for handwritten digit recognition. In: Fogelman-Soulié, F.; Gallinari, P. (eds.) Proceedings of the International Conference on Artificial Neural Networks Nanterre, France (1995)
38. Toh, A.; Togneri, R.; Nordholm, S.: Spectral entropy as speech features for speech recognition. In: Proceedings of PEECS2005, Perth (2005)
39. Togneri, R.; Toh, A.; Nordholm, S.: Evaluation and modification of cepstral moment normalization for speech recognition. In: Additibe Babble Ensemble Proceedings of the 11th Australian International Conference on Speech Science & Technology (2006)
40. Toh, A.; Togneri, R.; Nordholm, S.: Investigation of robust features for speech recognition in hostile environments. In: Asia-Pacific Conference on Communications, Perth (2005)
41. Kreβel, U.: Pairwise classifcation and support vector machines. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector Learning, Cambridge (1999)
42. Lefevre, F.; Gauvain, J.-L.; Lamel, L.: Genericity and probability for task-independent speech recognition. Comput. Speech Lang. **19**, 345–363 (2005)
43. Yang, J.; Zhang, D.; Frangi, A.; Yang, J.: Two-dimensional PCA: a new approach to appearance-based face representation and recognition. IEEE Trans. Pattern Anal. Mach. Intell. **26**(1), 131–137 (2004)
44. Dong, J.; Kryzak, A.; Suen, C.: A multi-net learning framework for pattern recognition. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, Seattle (2001)
45. Wright, J.; Yang, A.; Ganesh, A.; Sastry, S.; Ma, Y.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
46. Belongie, S.; Malik, J.; Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. **24**(4), 509–522 (2002)
47. Ergin, S.; Gerek, O.N.; Gülmezoğlu, M.B.; Barkana, A.: On the realization of common matrix classifier using covariance tensors. Digit. Signal Process. **41**, 110–117 (2015)
48. Teow, L.-N.; Loe, K.-F.: Robust vision-based features and classification schemes for off-line handwritten digit recognition. Pattern Recognit. **35**(11), 2355–2364 (2002)
49. Zhang, L.; Yang, M.; Feng, X.: Sparse representation or collaborative representation: which helps face recognition? In: International Conference on Computer Vision (ICCV-2011), Barcelona (2011)
50. Liu, C.-L.; Nakashima, K.; Sako, H.; Fujisawa, H.: Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recognit. **36**(10), 2271–2285 (2003)
51. Jiang, Z.; Lin, Z.; Davis, L.: Label consistent K-SVD: learning a discriminative dictionary for recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(11), 2651–2664 (2013)
52. Mayraz, G.; Hinton, G.: Recognizing handwritten digits using hierarchical products of experts. IEEE Trans. Pattern Anal. Mach. Intell. **24**(2), 189–197 (2002)