2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE).
Mexico City, Mexico October 25-27, 2023

2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) | 979-8-3503-0676-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCE60043.2023.10332834

Autoencoder with Orthogonal Variant

A.Anzueto-Ríos
Electrical Engineering Department
CINVESTAV-IPN
Mexico City, Mexico
alvaro.anzueto@cinvestav.mx

F. Gómez-Castañeda
Electrical Engineering Department
CINVESTAV-IPN
Mexico City, Mexico
fgomez@cinvestav.mx

L.M. Flores-Nava
Electrical Engineering Department
CINVESTAV-IPN
Mexico City, Mexico
lmflores@cinvestav.mx

J.A. Moreno-Cadenas
Electrical Engineering Department
CINVESTAV-IPN
Mexico City, Mexico
jmoreno@cinvestav.mx

***Abstract*—Autoencoder is a widely used neural architecture for dimensionality reduction. It can be considered similar to the principal component analysis (PCA) methodology. However, the final distribution of the components between classes does not establish orthogonality between them, which can result in a reduced separation between different classes. To address this issue, we present a modified autoencoder architecture with an orthogonal variant. The error minimization equation has been changed to ensure orthogonality between the final components. Our experiments show that the proposed orthogonal autoencoder architecture generates a final distribution with more separability than the PCA numerical process and a typical autoencoder architecture. This makes it a promising approach for applications that require high separability between different classes.**

***Index Terms*—principal component analysis, autoencoder, orthogonal process, neural network, and dimensionality reduction.**

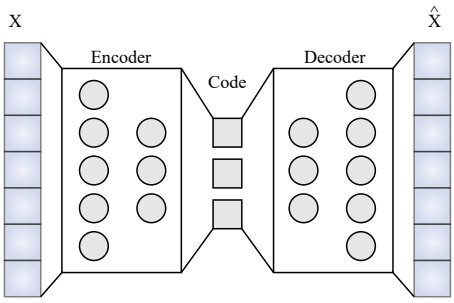


Fig. 1: Architecture of the Autoencoder.

I. Introduction

Data clustering is a complex topic that machine learning techniques have tackled. It contributes to a better understanding of the behavior of events from which samples formed by characteristics or traits are recorded. One can extract or identify patterns, trends, data groups with similar features, or atypical data from these samples. However, all this is done to analyze and interpret a larger goal, such as decision-making.

A database containing a high number of characteristics can be simplified using the Principal Component Analysis (PCA) technique [1], which is a data dimensionality reduction procedure. This new database is built with the important features gained from the mathematical process developed in PCA. However, the PCA method only involves the performance of a linear data projection while considering the variance. On the other hand, an autoencoder involves the development of a non-linear type of process that projects the data in more than one dimension, making it easier to extract the features that are the most representative of the problem space.

The autoencoder is a neural architecture in which the input vectors become the target vectors, and the error value is calculated by comparing the similarity of the input and output data [2]–[4]. The first component of an autoencoder is responsible for encoding the information at the input, while the second part is responsible for decoding, as shown in Fig. 1. The middle section (CODE) is responsible for data reduction and comprises the necessary numerical components with which the original data supplied at the input can be rebuilt. Reduced data, which are located in the CODE layer, can be put to use in the process of grouping or clustering, as described in [5]; this is the stage at which data groups that are distinguished by their similarities to one another are obtained, without taking into account any attribute that, in addition to grouping them, serves to differentiate them. The orthogonal autoencoder [6], which enhances the process of orthogonality of data, improving the separability between data groups, is a suggestion for data grouping and separation.

Thus, this paper presents the use of an autoencounter neural architecture with orthogonal-variant to process databases. The structure of the paper is as follows: In the next section, the methodology used to process three databases is presented; furthermore, each database is described. The following section describes and analyzes the results obtained for each case. The conclusions generated in this work are presented in the last section.

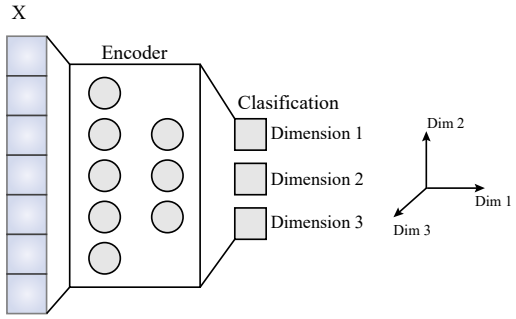


Fig. 2: Architecture of the Encoder.

II. Materials and Method

A. Architecture of Autoencoder

Autoencoders are neural architectures that are trained in an unsupervised manner, where the objective is to encode the vector of data that is presented in its input; this encoding gives a reduction of the original vector and, starting from their proceeds to the decording to reproduce the input vector from the learned encodings again. Therefore, the output of an autoencoder is the prediction of the neural architecture of the input data. The initial neural architecture (autoencoder) has been presented in Fig. 1; we start from left to right, where the input vector is represented by the vector x, which will be encoded into three components in the intermediate layer (also known as latency space), this compressed vector is identified by h. The vector h is decoded to produce the new vector $\hat{x}$ at the output. The vector $\hat{x}$ is the prediction or response of the network to the presence of vector x. Encoding (see Fig. 2) can be defined mathematically by (1), where W_{EN} are the synaptic weights and b_{EN} the polarization weights of the neurons in the input layer. Similarly, the decoding is represented in (2), where W_{DE} and b_{DE} are the synaptic and polarization weights, respectively. f and g represent the transfer functions associated with the neurons in the encoder and decoder layers. With the following definition for the dimension: $x, \hat{x} \in \mathbb{R}^n$ and $h \in \mathbb{R}^d$ and $W_{EN} \in \mathbb{R}^{d \times n}$ and $W_{DE} \in \mathbb{R}^{n \times d}$. In this work, the dimension $d = 3$ has been considered, which means that the distribution of the input data can be visualized in a graph, as depicted in Fig. 2.

$$h = f\left(W_{EN} \cdot x + b_{EN}\right) \tag{1}$$

$$\hat{x} = g\left(W_{DE} \cdot h + b_{DE}\right) \tag{2}$$

The loss function is presented in (3). It represents the weighted sum of the errors generated by the prediction of each sample when divided by the total number of samples contained in the database. This function is required for the autoencoder training process because it indicates how appropriate the reconstruction or prediction of the output vector concerns the input. The mean square error, defined by (4), is the error function.

$$L = \frac{1}{m} \sum_{j=1}^{m} e\left(x^{(j)}, \hat{x}^{(j)}\right) \tag{3}$$

$$e\left(x, \hat{x}\right) = \frac{1}{2} \left\|x - \hat{x}\right\|^2 \tag{4}$$

B. Architecture of Orthogonal Autoencoder

The training process for the autoencoder with the orthogonal variant is given by minimizing the loss function given in (5). h^T is the transpose of the matrix h, and I is the identity matrix. h has the dimension $d \times n$. A lambda factor is a weighting number of the orthogonalization term. It is determined that h provides orthogonality between the input vectors when the expression $h^T \cdot h$ tends to approach identity matrix I.

$$L = \left\|x - \hat{x}\right\|^2 + \lambda \left\|h^T h - I\right\|^2 \tag{5}$$

C. DataBases

In this work, three databases have been considered. The first one is a reference to a database known as the "IRIS database," which is frequently utilized for the task of classification and clustering. Iris Setosa, Iris Virginica, and Iris Versicolor are the three classes that can be found in the database [7]; each class has 50 samples. The length and width of both the sepal and the petal are measured in centimeters and are included as 4 of the characteristics of each sample.

The second database is referred to as the "Palmer Archipelago (Antarctica) penguin data," and Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, are the ones responsible for collecting the relevant data [8]. The database contains 344 penguin samples spanning three distinct species and three islands in the Palmer Archipelago in Antarctica. Four features were recorded for each sample collected, corresponding to the length of the flipper, the length and depth of the bill (each expressed in millimeters), and body mass (expressed in grams).

A chemical study of wines cultivated in the same region of Italy but derived from three different cultivars was used to compile the third database. The database contains 178 samples of three distinct types of wine; the 13 characteristics that were discovered in each of the three categories of wine are taken into consideration [9]. These characteristics are: 1—Alcohol; 2—Malicacid; 3—Ash; 4—Alkalinity of ash; 5—Magnesium; 6—Total phenols; 7—Flavanoids; 8—Nonflavanoid phenols; 9—Proanthocyanins; 10—Color Intensity; 11—Hue; 12—Diluted Wines; and 13—Proline.

Table I summarizes the parameters of the three databases.

TABLE I: Database Parameters

Name	Data	Feature	Classes	Dimensions
Iris	150	4	3	150×4
Penguins	344	4	3	344×4
Wine	178	13	3	178×13

III. Results and Discussion

The three databases have been run through three different dimensionality reduction procedures, the first of which was the numerical PCA approach, the second of which was the configuration of a conventional autoencoder (AE), and the third of which was the orthogonal variant in an autoencoder (OAE). All neural layers have been implemented with linear transfer functions, and a total of one thousand iterations have been carried out with the data set. 90% and 10% of the data have been used randomly for the training and validation phases, respectively. This division seeks that the model during the training process contains the largest number of points for each class, thus achieving a better approximation in the solution process of the orthogonal matrix. In the validation process, it is only verified that orthogonality has been achieved.

The Python programming language in its version 3.10 under the Spyder programming environment utilizing the Tensorflow library under the Windows 11 operating system with an i7 processor from the seventh generation was used to implement all three dimensionality reduction methods. Each database has its data preprocessed, and the normalization procedure is performed on each of the features of the data. Calculations based on the silhouette coefficient were performed to evaluate the quality of the data grouping.

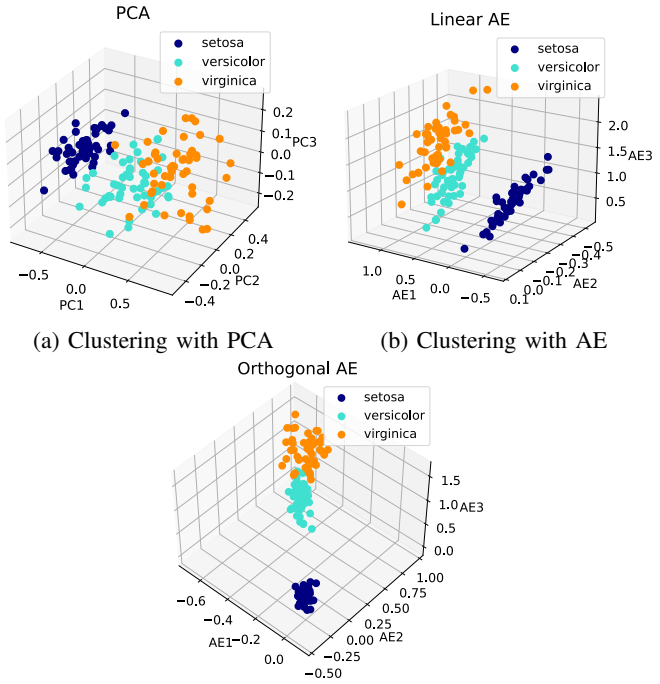
A. Processing using IRIS database

The Iris database was analyzed, and the suggested OAE method's results were compared with two benchmark methods, PCA and AE to determine their relevance. Fig. 3 shows the clusterings obtained by the three methods. In Fig. 3a, it was observed that the clusters obtained by the PCA method are the most dispersed; if we analyze the numerical scale on the axes of the 3D graphs, we can say that the distribution of the groups achieved by the OAE method is much more compact, see Fig. 3c.

Figure 4 presents the plots of the loss value calculated in the training and validation processes of AE and OAE. By analyzing the loss axis, it can be determined that for the training of the AE and OAE neural architectures, respectively, the loss value is less than 0.1 for both methods. Although the AE method achieves the minimum error with a lower number of iterations, it is relevant to mention that the OAE method consumes more computational resources when calculating the orthogonality between classes but manages to ensure that the groupings are compact.

B. Processing using PENGUINS database

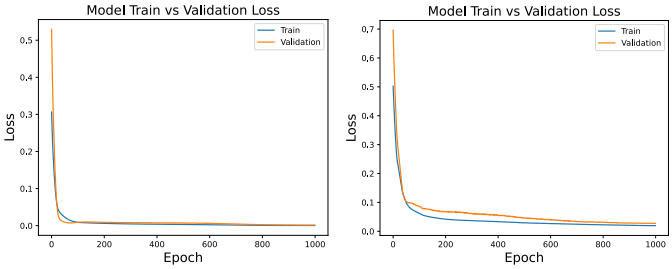
The three methods were again applied to the Penguin database, achieving clustering. The results show a similar behavior to the processing of the Iris database. The data distribution graphs are found in Fig. 5. The distribution that the OAE supplies is condensed and clearly defined, in contrast to the results shown by the PCA approach, which show a larger distribution of the data and result in the classes sharing spaces. The loss curves for the training and validation operations for the AE and OAE approaches are presented in Fig. 6. Analyzing



(a) Clustering with PCA (b) Clustering with AE

(c) Clustering with OAE

Fig. 3: Iris Database processing with three methods, it is important to determine the separability and compactness of the clusters in each method: (a) PCA; (b) AE; (c) OAE.



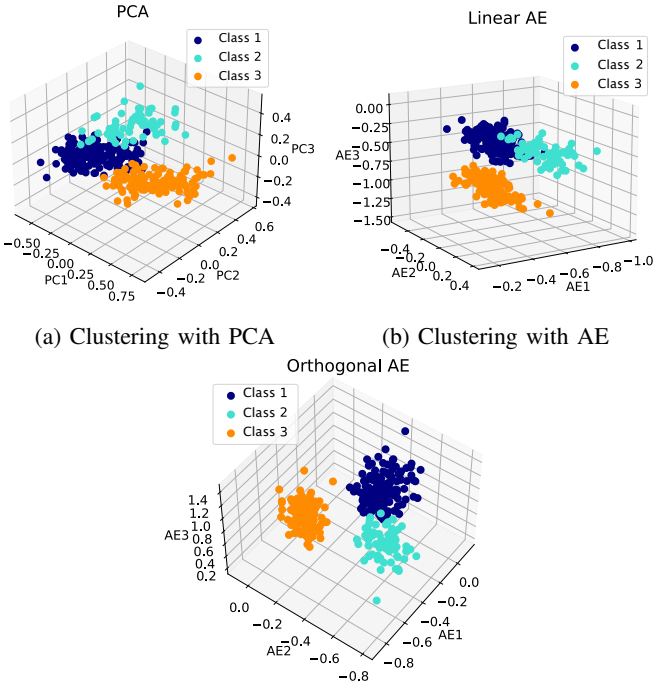
(a) Loss Graphics AE (b) Loss Graphics OAE

Fig. 4: Values of loss at each iteration in the training and data validation processes. (a)AE; (b)OAE.

the curves, it can be noticed that for both methods after iteration 200, the training and validation are similar; however, the loss value for AE is lower, although the clusters are more compact for OAE.

C. Processing using WINE database

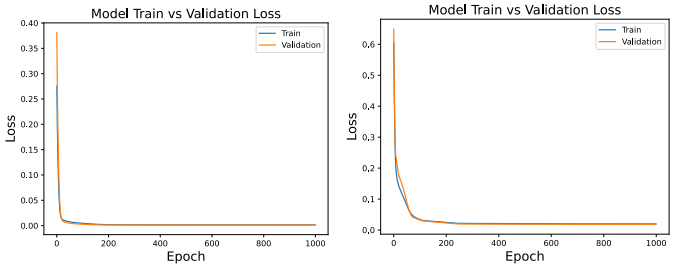
The wine database is complicated when you consider its 13 features and how closely they are related. Yet, the three algorithms produce good clustering, as seen in Fig. 7. This figure is well-known for having three groups corresponding to the expected classes. The loss plots for the training and validation phases are shown in Fig. 8 and are similar to those obtained for the previous databases; the loss values are less than 0.1.



(a) Clustering with PCA (b) Clustering with AE

(c) Clustering with OAE

Fig. 5: Penguins Database processing with three methods, it is important to determine the separability and compactness of the clusters in each method: (a) PCA; (b) AE; (c) OAE.

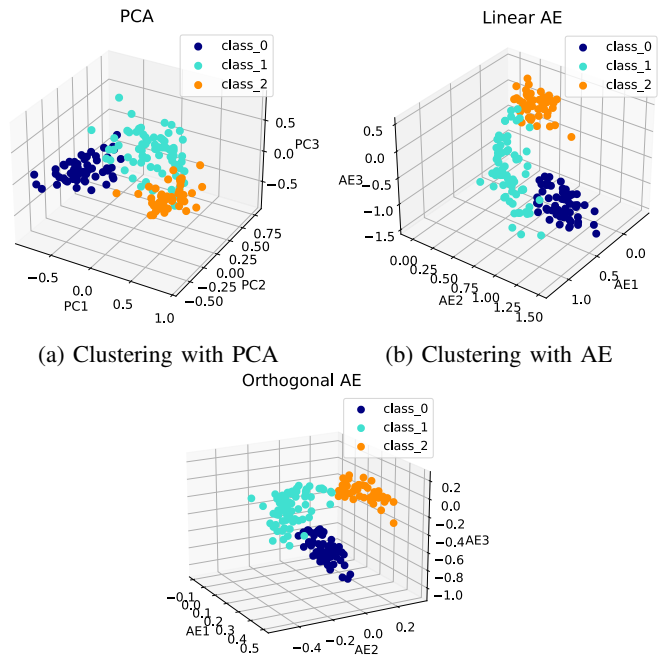


(a) Loss Graphics AE (b) Loss Graphics OAE

Fig. 6: Values of loss at each iteration in the training and data validation processes. (a)AE; (b)OAE.

Table II summarizes the data obtained from the silhouette coefficient [10] to determine the quality of the clusters for the three databases applying the three methods analyzed. It is important to call attention to the fact that the magnitudes of the axes in the data distribution graphs for the OAE result are significantly lower in comparison to those for the other approaches. This finding suggests that the clusters formed using OAE have reduced dispersion and improve the clustering capacity to be separated from one another. The best results are highlighted in bold. Analyzing these data, we can see that the orthogonal variant of the autoencoder presents the best data distributions for all cases.

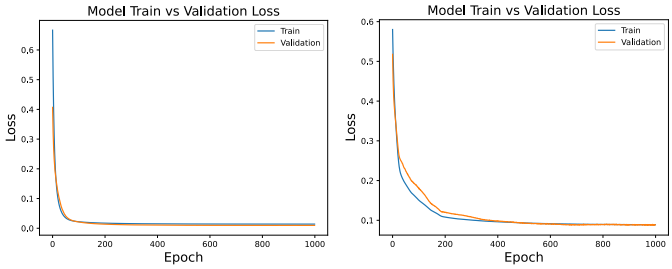
When examining the BigData paradigm and applying more robust approaches in data mining, such as the t-distributed



(a) Clustering with PCA (b) Clustering with AE

(c) Clustering with OAE

Fig. 7: Wine Database processing with three methods, it is important to determine the separability and compactness of the clusters in each method: (a) PCA; (b) AE; (c) OAE.



(a) Loss Graphics AE (b) Loss Graphics OAE

Fig. 8: Values of loss at each iteration in the training and data validation processes. (a)AE; (b)OAE.

TABLE II: Silhouette Coefficient

Database	PCA	AE	OAE
Iris	0.4265	0.5264	**0.6018**
Penguins	0.5011	0.5183	**0.6226**
Wine	0.4892	0.5506	**0.5941**

stochastic neighbor embedding (t-sne) method, it is possible to say that the orthogonal variation of the autoencoder can be utilized as a preprocessing when considering the analysis of these data. This perspective can be reached after analyzing the results of the study.

IV. Conclusion

The topic of data clustering has been tackled in this body of work, and in doing so, three distinct approaches have been investigated: the orthogonal variant, the numerical PCA, and the classic autoencoder. In terms of data clustering, the orthogonal variant displays superior quality, both in terms of compactness and in terms of separability between classes. The findings that were provided illustrate that class separation can be accomplished using any one of the three approaches. The silhouette coefficient is obtained in order to conduct an analysis of the quality of clustering and separability. This analysis demonstrates that the orthogonal variant is the methodology that most effectively completes the task. The orthogonal variant of the autoencoder can be used as a preprocessing of the data in order to apply more robust methods in data mining. One example of this would be the t-distributed stochastic neighbor embedding method, which can provide solutions to emerging issues such as the Internet of Things (IoT) and machine learning.

References

[1] S. Ji, Z. Wuang, and Y. Wei, "Principal component analysis and autoencoders," available online: http://people.tamu.edu/ sji/classes/PCA.pdf (accessed on 22 June 2023).

[2] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[3] E. Santana, M. Emigh, and J. C. Principe, "Information theoretic-learning auto-encoder," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 3296–3301.

[4] S. Lee and J. Jo, "Information flows of diverse autoencoders," *Entropy*, vol. 23, no. 7, p. 862, 2021.

[5] A. Anzueto-Ríos, F. Gómez-Castañeda, L. Flores-Nava, and J. Moreno-Cadenas, "Metaheuristic method for dimensionality reduction tasks," in *2022 19th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2022, pp. 1–5.

[6] X. Wang, Y. Lu, X. Lin, J. Li, and Z. Zhang, "An unsupervised classification algorithm for heterogeneous cryo-em projection images based on autoencoders," *International Journal of Molecular Sciences*, vol. 24, no. 9, p. 8380, 2023.

[7] U. of California, "Iris data set," available online: https://archive.ics.uci.edu/dataset/53/iris (accessed on 26 June 2023).

[8] A. M. Horst, A. P. Hill, and K. B. Gorman, *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020, r package version 0.1.0. [Online]. Available: https://allisonhorst.github.io/palmerpenguins/

[9] U. of California, "Wine data set," available online: https://archive.ics.uci.edu/dataset/109/wine (accessed on 28 June 2023).

[10] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.