Decoding Meaning from Data: Utilizing Autoencoders for enhanced Pattern Recognition

Godfrey Bryan Satiada, Nico Rafael Ting, Gregory Uy

ASITE, Asian Institute of Management, Manila, Philippines

Abstract

This study explores the use of autoencoders as a data preprocessing step to improve feature separation by latent factor extraction. Data Compression through various methods is useful in Data Science to improve feature separation or as feature engineering or vectorization. Our study furthers this by attempting to make a universalizable method for preprocessing by using a proximity matrix and then an autoencoder to use relationships between rows as the primary features that we extract and use for clustering. Clustering is done with K-means and Ward's clustering algorithms, and the best performing results are noted. We evaluate this method over five University of Irving public datasets commonly used as benchmarks and show in which cases that the method can effectively improve feature extraction and thus the clustering results.

Keywords: Autoencoder, Clustering, CNN, Proximity Matrix, Dimension Reduction

Introduction

Data Preprocessing and Cleaning is a necessity to improve the results of any machine learning method. It can vary from simply removing invalid or empty values or be as complex as feature engineering with domain expertise to intentionally improve the results of later steps. Common data preprocessing steps include dropping invalid rows, imputing missing values, feature engineering additional features, data vectorization, or dimensionality reduction.

Various articles [5], [3], [1], had shown that by using different data preprocessing techniques could improve the clustering metric score of different clustering techniques. Some of these preprocessing ranges for dimensionality reduction [1] up to the state-of-the-art Neural Networks [5], [3]. Of course, different processing techniques had garnered different results for instance the best result got in [1] with MNIST and USPS dataset had garnered an accuracy score of 0.9054 and 0.9584 respectively while [5] got the best accuracy score of 0.882 and 0.926 respectively for the same dataset.

Modern machine learning models generally use a pipeline of data preprocessing and transformation steps for normalization, feature engineering, vectorization, and even multiple layers of neural networks. Our study is interested in the use of autoencoders as a replacement for other preprocessing

steps to extract latent features from our data through dimensionality reduction. In training a neural network to encode and decode the input rows of our data, the encoder is trained to create a compressed representation of our data. This is confirmed by the decoder if it can recreate the original input feature columns as it shows that the information of the original data was effectively retained by the compressed representation. A similarity matrix of the original data is used to extract universal relationships throughout the rows, and to increase the number of columns for the autoencoder to effectively extract features. The autoencoder architecture that we use for the study is a combination of convolutional and pooling for the encoder, and up sampling for the decoder. This combination of preprocessing and autoencoder architecture should be applicable for any kind of data, and especially extract datapoint relationships and improve feature separation.

Autoencoder is an unsupervised learning with the goal to find the best representative of a data by reconstructing it [2], [6], [8]. Autoencoder is made up of neural networks and is generally represented into two parts the encoder and decoder. The encoder accepts the input, reconstructs the input and represents it in low dimensional representation [6] in other words, encoder acts as the dimensionality reduction of the autoencoder. On the other hand, decoder accepts the low dimensional representation as the input and reconstructs that input wherein it approximates the original input as an output. This output is approximation of the original input since it tries to remove or reduce the noise of the original input. This output could have different applications such as anomaly detection, signal processing, and image and video processing [7], [4].

This study then uses the encoded and decoded data in comparison with the raw data for an unsupervised machine learning problem to give a baseline performance for our method that is not based on the ability of our machine learning model to learn and infer relationships in the data but is based on the separation and improvement in data interpretability. For this, we use clustering algorithms to cluster the results of the raw, encoded, and decoded datasets then evaluate them based on their cluster purity to see if there is improvement in data separation based on the known labels.

Data and Methodology

Five Datasets from the University of California Irving Machine Learning Repository were used for this study to cover a wide variety of data distributions. All the data sets used are appropriate for clustering algorithms and the description is shown in Table 1.

Dataset Name	Description
Iris	Iris Flower Variety
Wine	Cultivar of Grapes in the same region
Breast Cancer Wisconsin (Original)	Breast Cancer Metastasis in patients
Glass Identification	Glass Identification for Crime Scene Investigation
Yeast	Yeast Cellular Localization Sites in the body

Table 1. Different datasets used for the experiment

Proximity Matrix

Proximity Matrix can come into two types: a distance-based or similarity-based matrix. The purpose of creating a proximity matrix is to limit the possible patterns that we can extract from the data. This pattern may come into the form of how far/near the data points from each other. Since we are dealing with clustering techniques, it is appropriate to use this proximity matrix as we are explicitly saying which group can be formed from the matrix.

Autoencoder

The Autoencoder for this project was inspired by the CNN architecture. We have used multiple layers of convolution and max pooling to extract the important features of each row in the similarity matrix. The reason behind this architecture is to extract the bigger distance/similarity to formally create a gap of separation among the data points, once this separation between the datapoints is visible the clustering would have less of difficulty.

In this process, we are expecting two different data the encoded data which is the output of the encoder process and the decoded data which is approximation of the original input. The autoencoder will be trained on the proximity matrix with an optimizer of Adam and loss function of Mean Squared Error (MSE).

Cluster

Finally, once we got our data from the encoded, decoded, and as well as the raw data, we would like to use clustering techniques on these datasets to see if there is an improvement on the clustering metrics scores. For this experiment we had employed two kinds of clustering techniques: K-Means and Agglomerative: Ward's clustering. The two methods were the usual choice when doing clustering, hence the reason we chose them. Moreover, the metric score that we will be using are Purity, Adjusted Rand Index (ARI), and Adjusted Mutual Information (AMI). These scoring metrics are external metrics meaning we have a ground truth label for comparisons.

Pseudo code for the whole pipeline
1 Input any dataset
2 Proximity Matrix -> Dataset
3 Autoencoder
4 Encoder -> Proximity Matrix
5 Convolution
6 Max Pooling
7 Output -> Encoded Data
8 Decoder -> Encoded Data
10 Convolution
11 Up sampling
12 Output -> Decoded Data
13 Clustering on dataset, encoded data, and decoded data
14 Evaluate the clustering result

Results and Discussion

Before looking at the clustering metrics score, it would be best to look at the different scatter plot given by the raw, encoded, and decoded data.

Raw Data

Figure 1a. Scatter plot of Raw Wine dataset

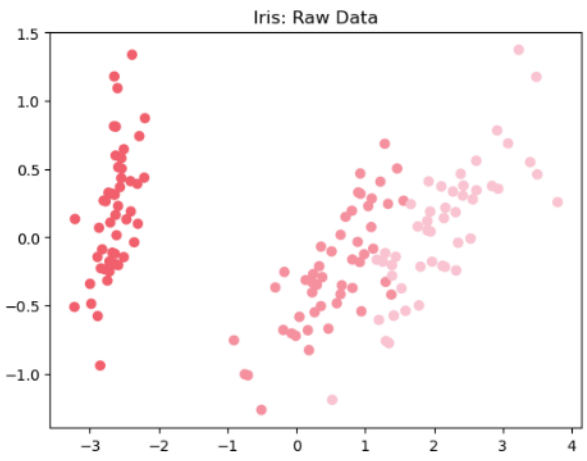
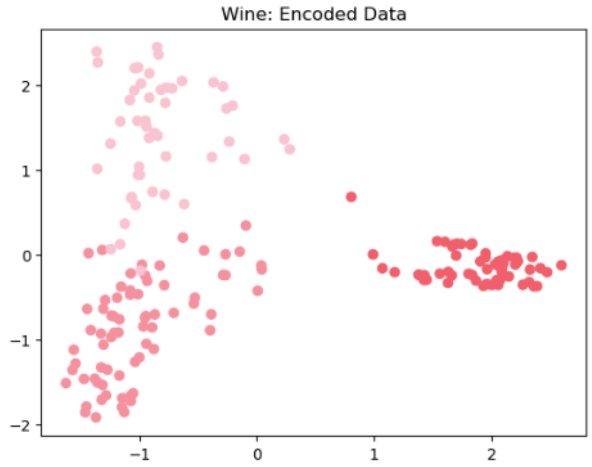


Figure 1b. Scatter plot of Raw Iris Dataset

Encoded Data



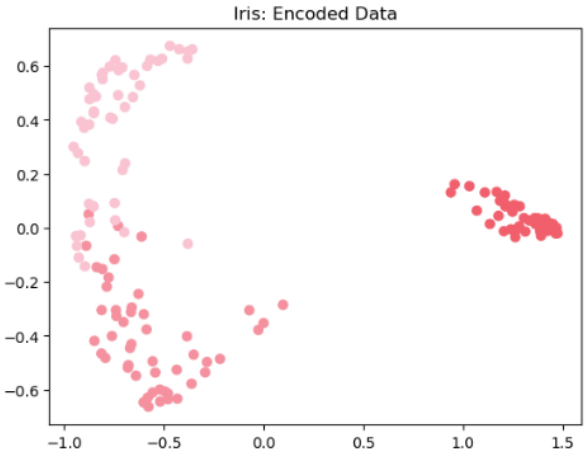


Figure 2a. Scatter plot of Encoded Wine dataset

Figure 2b. Scatter plot of Encoded Iris Dataset

Decoded Data

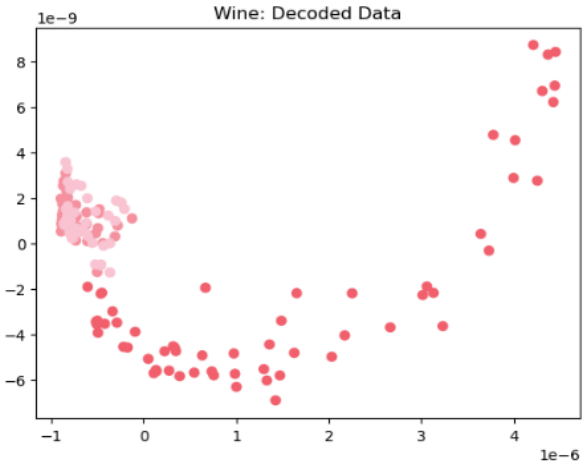


Figure 3a. Scatter plot of Decoded Wine dataset

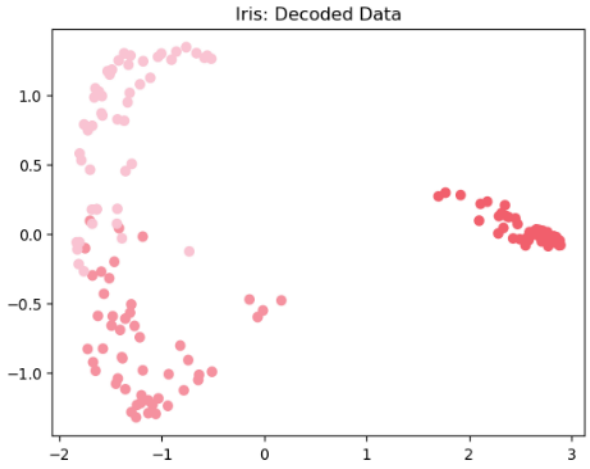


Figure 3b. Scatter plot of Decoded Iris Dataset

Principal Component Analysis is used to create a visualization of the data sets and were labeled with the original dataset target labels. Looking at the different figures we can really see that the autoencoder was able to separate the datapoints clearly. From figure1a, we can clearly see that there are still some that are overlapping the clustering, or the separation of data points are not visible by the eye, yet when it gone the encoder (figure2a) we can see that the overlapping of datapoints were almost negligent however as it was decoded the datapoints became messier. The same scenario is happening with Iris dataset (figure 1b, 2b, and 3b), we can see that in figure 1b, there are points that overlap, and it was minimized when it undergone encoding and decoding.

Clustering Metric Scores

K-Means	Decoded Data			Encoded Data			Raw Data		
Data	purity	AMI	ARI	purity	AMI	ARI	purity	AMI	ARI
wine	63.48%	37.96%	22.16%	75.84%	69.90%	55.40%	68.53%	41.60%	35.18%
iris	96%	86.05%	88.57%	97.33%	89.99%	92.21	89.33%	75.51%	73.02%
yeast	44.20%	16.90%	10.02%	45.20%	18.62%	8.50%	51.28%	24.75%	14.11%
Glass	53.27%	35.69%	22.31%	54.67%	35.97%	20.89%	53.74%	40.50%	24.64%
BC (original)	92.24%	66.86%	71.28%	92.39%	67.24%	71.77%	96.05%	74.75%	84.65%
BC (Diagnostic)	65.72%	5.62%	4.55%	89.80%	54.50%	63.29%	85.41%	46.40%	49.14%

Table 2. Clustering Metric Scores of K-Means

Ward's	Decoded Data			Encoded Data			Raw Data		
Data	purity	AMI	ARI	purity	AMI	ARI	purity	AMI	ARI
wine	65.73%	42.67%	27.17%	92.67%	83%	80.33%	69.66%	40.99%	36.84%
iris	91.33%	81.15%	77.34%	90%	79.54%	74.55%	89.33%	76.72%	73.12%
yeast	42.92%	16.90%	9.32%	46.70%	19.75%	9.96%	48.59%	22.03%	10.71%
Glass	62.15%	34.17%	21.16%	62.15%	33.32%	22.63%	54.21%	36.51%	26.20%
BC (original)	94.44%	73.35%	78.89%	96.20%	76.31%	85.24%	96.63%	80.28%	86.90%
BC (Diagnostic)	65.73%	5.62%	4.56%	68.19%	6.18%	10.52%	77.86%	31.80%	28.72%

Table 3. Clustering Metric Scores of Ward's Method

Table 2 and 3 showed us different metric scores per clustering and per dataset. In K-Means (table 2) we can see a significant improvement of clustering from raw data to encoded data, although not all datasets had this improvement, their scores are not far from one another. While on Ward's (table 3) we can clearly see a significant improvement, especially with Wine dataset's purity score. Although fifty percent of the dataset had the highest score when the clustering was used in the raw data. The rest of the dataset performed well when it was encoded and/or decoded.

When the initial datasets already represent all the important features to make accurate predictions, such as in simple data sets or those data sets that can be very accurately predicted even with simple models, this method does not add to the results and can instead act as confounding variables and unnecessarily increasing the complexity. In data sets that are missing some latent features and where the target features are truly separate from latent features such as the wine data set, our method is very effective at identifying and creating feature separation in the compressed representation. However, in the last case where the system is extremely complex or where the feature columns themselves do not lend easily to data interpretation such as in the yeast data set, the method may itself need preprocessing to prepare the data to extract effective features.

Conclusion and Recommendations

Undeniably data processing will always be part of the data scientist's pipeline. Finding the right preprocessing technique for a certain dataset is a crucial step for extracting all possible insights and

outputs a dataset can give. On the other hand, if we can generalize this data processing into one pipeline wherein it will always give us the ideal result then that would be great. However, such a thing does not exist yet. One instance is our Autoencoder pipeline, though it was able to improve results of some datasets, it still cannot accommodate all possible datasets there are.

The Autoencoder pipeline presented in this paper follows a neural network pattern finding where we want to retain the most important features of a datapoints. Having said that, a neural network pattern finding doesn't necessarily solve everything, for instance a dataset where the patterns are already explicit, if we apply our autoencoder pipeline to this, wouldn't that just make the dataset more complex?

Finally, just like any other methodologies and techniques this autoencoder is not perfect and could be further improved. An exploration of different neural networks for encoding and decoding could give us more robust data processing techniques. In addition to that, the proximity matrix might not be, in some cases, needed, hence we recommend using different techniques such as feature engineering. Lastly, we would like to recommend exploring more different clustering techniques as it might produce a more accurate result.

References

Articles from the UCI Machine Learning Repository:

- Aeberhard, S., & Forina, M. (1991). Wine. UCI Machine Learning Repository. https://doi.org/10.24432/C5PC7J.
- Fisher, R. A. (1988). Iris. UCI Machine Learning Repository. https://doi.org/10.24432/C56C76.
- German, B. (1987). Glass Identification. UCI Machine Learning Repository. https://doi.org/10.24432/C5WW2P.
- Nakai, K. (1996). Yeast. UCI Machine Learning Repository. https://doi.org/10.24432/C5KG68.
- Wolberg, W. (1992). Breast Cancer Wisconsin (Original). UCI Machine Learning Repository. https://doi.org/10.24432/C5HP4Z.

Journal Articles and Conference Proceedings:

(1) Allaoui, M., Kherfi, M. L., & Cheriet, A. (2020). Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study. In: El Moataz, A., Mammass, D.,

Mansouri, A., Nouboud, F. (eds) Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science, vol 12119. Springer, Cham. https://doi.org/10.1007/978-3-030-51935-3_34.

(2) Baldi, P. (2012, June). Autoencoders, unsupervised learning, and deep architectures. In Proceedings of ICML workshop on unsupervised and transfer learning (pp. 37-49). JMLR Workshop and Conference Proceedings.

(3) Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., & Cui, P. (2020). Structural Deep Clustering Network. In Proceedings of The Web Conference 2020 (WWW '20). Association for Computing Machinery, New York, NY, USA, 1400–1410. https://doi.org/10.1145/3366423.3380214.

(4) Chiang, H. T., Hsieh, Y. Y., Fu, S. W., Hung, K. H., Tsao, Y., & Chien, S. Y. (2019). Noise reduction in ECG signals using fully convolutional denoising autoencoders. IEEE Access, 7, 60806-60813.

(5) Gultepe, E., & Makrehchi, M. (2018). Improving clustering performance using independent component analysis and unsupervised feature learning. Human-centric Computing and Information Sciences, 8, 25. https://doi.org/10.1186/s13673-018-0148-3.

(6) Michelucci, U. (2022). An introduction to autoencoders. arXiv preprint arXiv:2201.03898.

(7) Pawar, A. (2020, December). Noise reduction in images using autoencoders. In 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 987-990). IEEE.

(8) Tschannen, M., Bachem, O., & Lucic, M. (2018). Recent advances in autoencoder-based representation learning. arXiv preprint arXiv:1812.05069.