AutoEncoders

Mouhamed El Mamoune DIEYE, Phanie Dianelle Negho, Regis Konan Marcel DJAHA

April 11, 2023

African Master's in Machine Intelligence (AMMI)

1 Introduction

Autoencoders are a type of neural network that has become increasingly popular in recent years due to their ability to learn efficient data representations without the need for labeled data. It consists of two parts: an encoder and a decoder. The encoder takes an input and produces a compressed representation of it, while the decoder takes this compressed representation and produces a reconstruction of the original input.
Autoencoders is an unsupervised learning model since the loss function we have to minimize is between the input and the output.

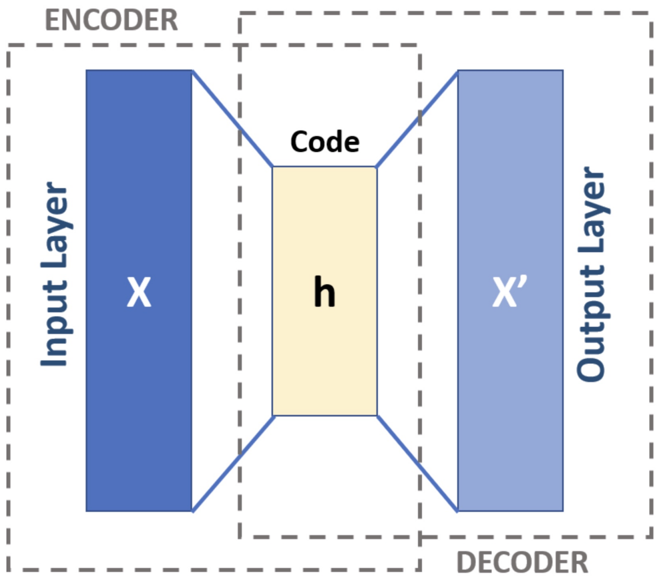

Figure 1: General structure of an autoencoder

The objective of autoencoders is to minimize the reconstruction error between the original input data and its reconstructed output, while also minimizing the size of the compressed representation.

By doing so, autoencoders can learn a compressed representation of the input that can be used for tasks such as dimensionality reduction, feature extraction, or anomaly detection.

There exists many type of autoencoders according to the architecture. In the case of simple autoencoders, we can talk about undercomplete and overcomplete autoencoders. An autoencoder is called undercomplete if the dimensions of the hidden layers are smaller than the dimension of the input and output layers. As far we progress from the input (output layer) to the bottleneck, the dimension of the hidden layers decreases. This is the architecture that we will use for the work. An autoencoder is call overcomplete if the dimensions of the input and output layers are smaller than the dimensions of the hidden layers. As far as we progress from the input layer(output layer) the number of units in the hidden layers increases.

The rationale behind the use of an undercomplete autoencoder Figure 2 is that to train the model to be able to capture the most salient informations of the input with a low dimensional representation (bottleneck). This is like performing dimension reduction while preserving the most useful information of the input so that we can reconstruct with the highest possible precision. This dimensionality reduction is crucial in the case where we have high dimensional data that might require much resources of computation and a high complexity for the data manipulation.

Autoencoders have a wide range of applications, including image and speech recognition, anomaly detection, natural language processing, and recommendation systems. They are particularly useful in cases where there is no labeled data available for training, as they can learn useful features from raw data in an unsupervised manner.

2 Data and Mathematics of autoencoders

2.1 Data

We will use the extremely popular MNIST dataset as input. It contains black-and-white images of handwritten digits.

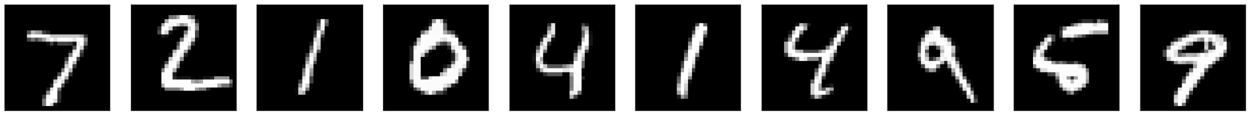

Figure 2: Original images

They're of size 28×28 and we use them as a vector of 784 numbers between $[0, 1]$. Check the `https://github.com/RegisKonan/FinalProject/blob/551359d22905fcbdab9a889b2bcb6ae12c9257e2/Autoencoder.ipynb` for the details.

2.2 Mathematics of autoencoders

The mathematical intuition behind autoencoders is to find two functions f and g such that : $X \in \mathrm{R}^m$, $h \in \mathrm{R}^p$, so , $p < m$

$$\begin{aligned} f : \mathrm{R}^m &\longrightarrow \mathrm{R}^p \\ g : \mathrm{R}^p &\longrightarrow \mathrm{R}^m \\ g(f(X)) = \hat{X} &\in \mathrm{R}^m \end{aligned} \tag{1}$$

The mathematical definition of an autoencoder suggests here that the functions f and g are respectively the encoder and the decoder. The output of the function $f(X) = h$ is the representation of the bottleneck.

2.2.1 Loss Function

The goal of a model is to minimize the error between the input and the output, this means we have to minimize the distance between our input and output using a defined loss function. In our case, we can derive two loss functions according to the magnitude of our inputs. We are using MSE if our input are real values and BCE if our data are either 0 or 1.

- **Mean square error**

$$\begin{aligned} L(X, \hat{X}) &= \frac{1}{M} \sum_{i=1}^{M} ||X_i - \hat{X}_i||^2 \\ L(X, \hat{X}) &= \frac{1}{M} \sum_{i=1}^{M} ||X_i - g(f(X_i)||^2 \end{aligned} \tag{2}$$

where M is the number of observations in our training dataset
In other to minimize our loss function, let us calculate the derivative of MSE with respect to a specific observation j.

$$\frac{\partial L}{\partial \hat{x_j}} = 0 \tag{3}$$

so, by assuming that the inputs are one-dimensional and taking a specific observation j we get:

$$\frac{\partial L}{\partial \hat{x_j}} = -\frac{2}{M}(x_j - \hat{x_j}) \tag{4}$$

(3) is satisfied when $x_j = \hat{x_j}$, we assume that, $\frac{\partial^2 L}{\partial \hat{x}_j^2} > 0$, therefore, we can conclude that $x_j = \hat{x_j}$ is the optimal minimum of the MSE.

- **Binary Cross-Entropy**
  If the activation function of the output layer of the FFA is a sigmoid function, and we normalized our input to be between 0 and 1.

$$L_{CE} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{M}(x_{j,i}log\hat{x}_{j,i} + (1 - x_{j,i})log(1 - \hat{x}_{j,i}) \tag{5}$$

  where $x_{j,i}$ is the j^{th} feature of the i^th observation. The sum is over the entire set of observations and overall components of the vectors.

  let us consider the simplified case where x_i, $\hat{x}_i$ and are one-dimensional to make our thing easier and do the same we did for the MSE.

$$\frac{\partial L_{CE}}{\partial \hat{x}_i} = 0 \tag{6}$$

  In this case, it is easy to show that the binary cross-entropy LCE is minimized when $x_i = \hat{x}_i$.

  To find when the L_{CE} is minimized we can derive L_{CE} with respect to a specific input x_j

$$\begin{aligned}
\frac{\partial L_{CE}}{\partial \hat{x}_j} &= -\frac{1}{M}\left[\frac{x_j}{\hat{x}_j} - \frac{1 - x_j}{1 - \hat{x}_j}\right] \\
&= -\frac{1}{M}\left[\frac{x_j(1 - x_j) - \hat{x}_j(1 - x_j)}{\hat{x}_j - \hat{x}_j{}^2}\right] \\
&= -\frac{1}{M}\left[\frac{x_j - \hat{x}_j}{\hat{x}_j - \hat{x}_j{}^2}\right]
\end{aligned} \tag{7}$$

  Since we tried to satisfy (7) , this can be possible if only if $x_j = \hat{x}_j$, we assume that $\frac{\partial^2 L_{CE}}{\partial \hat{x}_j{}^2} > 0$ to ensure this is a global minimum.

3 Results and Discussions

We will now implement the autoencoder with Pytorch. The hyperparameters in Figures 3, 4 and 5 for the encoder are: 128 units in the hidden layer, the bottleneck dimension is 64. The activation function of the output layer is sigmoid and MSE is the loss function for Figure 3. For Figure 4, we used ReLU as activation function and MSE as Loss function. For 5, we used Sigmoid and BCELoss respectively for activation and loss functions.

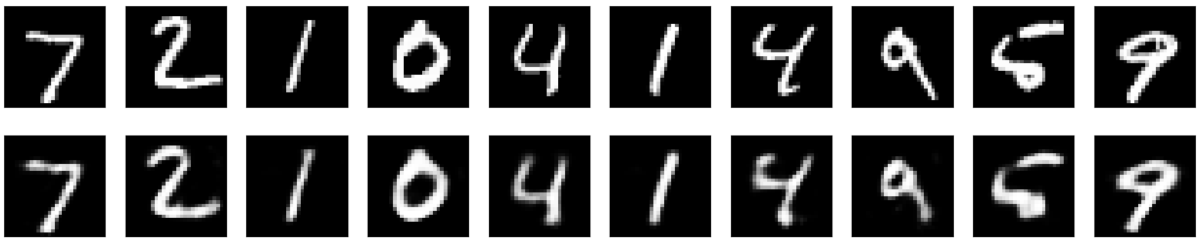
Figure 3: autoencoder trained with sigmoid and MSE

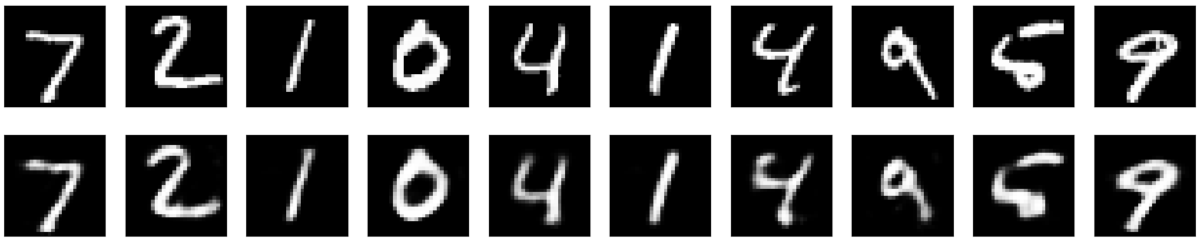
Figure 4: autoencoder trained with ReLU and MSE

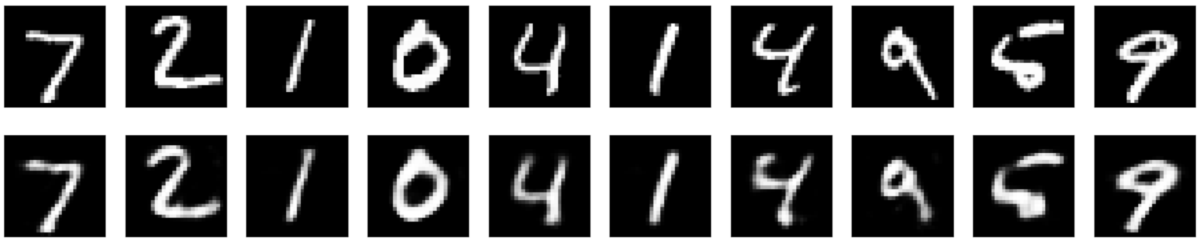
Figure 5: autoencoder trained with sigmoid and BCE

For the 3 figures mentioned above, the reconstruction of the input is almost similar in the 3 cases, and also the results are impressive compared to the original images. This means that the models are highly able to reconstruct the input with a high precision.

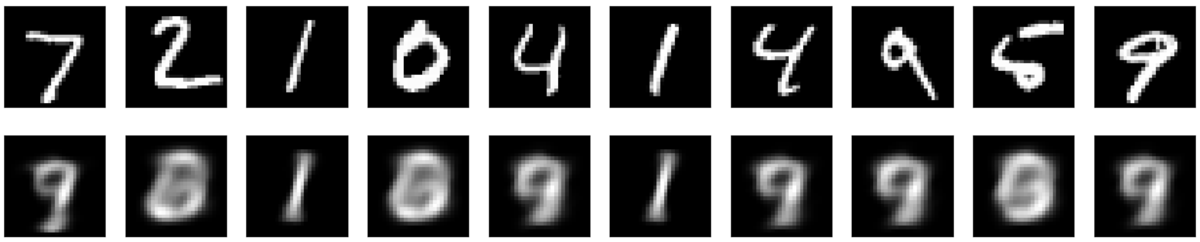
Figure 6: autoencoder trained with sigmoid and BCE (reduction of the number of units)

In the Figure 6, the autoencoder is trained using 8 units in the hidden layer and the bottleneck dimension is 4. We also used sigmoid and MSE respectively for activation in the output layer and loss functions. The result shows blurred images, this is due to the fact that the dimensions are strongly reduced. We can assume that as long as we reduce the hidden layers dimensions (number of units), there is no guaranty that the model will be able to capture the most salient information from the input. Thus, reducing the model's hyperparameters to the extreme is no use because the most salient information of the input will not be captured in order to do a good reconstruction.

4 Conclusion and Recommandations

In this work, we investigated the simple autoencoder in unsupervised Learning. This study involves the notions of undercomplete, overcomplete, architecture of autoencoder, Compute gradients using back-propagation, minimizing of the reconstruction error, and some activations functions such as Mean square error and Binary Cross-Entropy.

We implemented the simple autoencoder using both MSE and BCE and knowing that the Bottleneck is the most important part of our model, we also played with its dimensionality and observed that, when we compressed a lot of our model, we can lose information of the input. To go further, variational autoencoders such as the variational autoencoder, the Sparce autoencoder, and the contractive autoencoder can be introduced.

References

[1] Umberto Michelucci (2022) *An Introduction to Autoencoders* , TOELT.AI.

[2] Alireza Makhzani (2018) *Unsupervised Representation Learning with Autoencoders*, Ph.D. thesis, University of Toronto.

[3] William L. Hamilton (1994) *Fundamentals of Machine Learning, Lecture 25 — Autoencoders and self-supervision.*