Icon

Description automatically generated(Sparse) Autoencoders

In this chapter we will look at autoencoders, and in particular to sparse autoencoders. This is a theory chapter, so it will cover the mathematics and the fundamentals of autoencoders. We will discuss what they are, what are the limitations, the typical use cases and we will look at some examples.

# What is an autoencoder

As we have seen in the many previous chapters neural networks are typically used in a supervised setting. Meaning that for each training observations we will have a label or expected value . The neural network model will then try to learn the relationship between the input data and the expected values. Now suppose we have only unlabeled observations, meaning we only have our training dataset , made of the observations with

Where in general, as noted previously in the book, with some integer. In general, we can give a slightly intuitive definition of an autoencoder as

Definition: an autoencoder is a type of neural network that tries to learn to reconstruct the input observations with the lowest error possible*[[1]](#footnote-1)*.

In other words, you are training a neural network to learn the identity function. At this moment you may wonder why learning the identity function could be useful. This will become clear immediately in the next sections.

# Feed Forward Autoencoders

A Feed Forward Autoencoder (FFA) is a neural network made by dense layers. In Figure (25.1) you can see an example of an FFA.



Figure 25.1: This is a typical architecture of a Feed Forward Autoencoder. The number of neurons in the layers at first goes down as we move through the network until it reaches the middle, and then starts to grow again until the last layer has the same number of neurons as the input dimensions.

A typical architecture has an odd number of layers. Typically, the first layer has a number of neurons . Moving toward the center of the network the number of neurons in each layer drops in some measure. The middle layer (remember we have an odd number of layers) is normally the one with less neurons of all. In almost all practical applications the layers after the middle one, are a mirrored version of the layers before the middle one. For example an autoencoder with 3 layers could have the following numbers of neurons: , and then (supposing we are working on a problem where the input dimension ). All the layer up to and including the middle one are called the **encoder**, and all the layers from and including the middle one (up to the output) are called the **decoder**, as you can see depicted in Figure (25.1).

The **encoder** can be written generally as a function

Where is the output of the middle layer in Figure (25.1) when applied to the input observation . Note that with typically . Also, relevant to point out is that could even be one or two orders of magnitude smaller than . The **decoder can be written as**

If the FFA is successful and then the decoder is able to reconstruct the input by using only much smaller number of features () than the input observations originally have ().

**Note** The **encoder** is able to reduce the number of dimensions of the input observation and create a learned representation of the input that has a much smaller number of dimensions. This learned representation is enough for the **decoder** to reconstruct the input accurately (if the autoencoder works as intended).

To give the reader a concrete impression of how powerful those neural networks are let’s consider two examples.

## Reconstruction of hand-written digits

Let’s consider the MNIST dataset. This dataset, as you should know by know at this point in the book, contains hand-written digits from 0 to 9. Each image is pixels, that means that we have 784 features (the pixel gray values) as inputs. But how many features are really necessary to reconstruct the images? Let’s try to answer this question. Let’s start with an autoencoder with 3 layers with the numbers of neurons in each layer equal to . Note that the first and last layers **must** have a dimension equal to the input dimensions. For this example, we used the Adam optimizer, as loss function (we will discuss that more in detail later) the cross-entropy and we trained the model for epochs with a batch size of . In Figure (25.2) you can see two lines of digits from the MNIST dataset. The line at the top contains the original images, while the one at the bottom are the reconstructed images with the autoencoders just described.

A picture containing sign

Description automatically generated

Figure 25.2: In the top line you can see the original digits from the MNIST dataset. While the line below are the digits reconstructed by the autoencoder with number of neuraons equal to (784, 16, 784).

It is impressive that to reconstruct an image with 784 pixels only 16 features are needed to have a result that, although not perfect, allows us to understand almost perfectly what digits was used as input. Increasing the size of the middle layer to (and leaving all other parameters the same) gets a much better result as you can see in Figure (25.3).

A picture containing text

Description automatically generated

Figure 25.3: In the top line you can see the original digits from the MNIST dataset. While the line below are the digits reconstructed by the autoencoder with number of neuraons equal to (784, 64, 784).

This tell us that the information of the images is really contained in a lower number of features than 784.

**Note** An autoencoder with a middle layer smaller than the input dimensions can be used to extract the important features of an input creating a learned representation of the inputs given by the function . Effectively an FFA can be used to perform dimensionality reduction.

The FFA will not be able to recreate the input digits well, if the number of neurons in the middle layer is reduced too much. In Figure (25.4) you can see the reconstruction of the same digits with an autoencoder with only 8 neurons in the middle layer.

A close up of a sign

Description automatically generated

Figure 25.3: In the top line you can see the original digits from the MNIST dataset. While the line below are the digits reconstructed by the autoencoder with number of neuraons equal to (784, 8, 784).

In Figure (25.4) you can see a comparison of the reconstructed digits by all the FFAs we have discussed.



Figure 25.4: In the top line you can see the original digits from the MNIST dataset. The second line of digits are the digits reconsructed by the FFA (784,8,784), the third by the FFA (784,16,784) and the last one by the FFA (784,64,784).

From Figure (25.4) you can clearly see how, increasing the size of the middle layer, the reconstruction is better and better.

# Autoencoders Applications

## Dimensionality Reduction

Comparison with PCA, batches, higher dimensions

## Classifications

Using latent representation to do classification in higher dimensions

Calendar

Description automatically generated

Points to remember:

Running time (between 784 features and 8 for example)

Comparison of accuracy

### Curse of dimensionality in kNN

https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02\_knn\_notes.pdf

Curse of dimensionality

<https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html>

## Anomaly Detection

Using the reconstruction error

# Variational Autoencoders

1. In this chapter we will discuss at length what we mean with error here. [↑](#footnote-ref-1)