Introduction to Autoencoders

1 Introduction

An autoencoder is a neural network that is trained to copy its input to its output, with the typical purpose of dimension reduction - the process of reducing the number of random variables under consideration. It features an encoder function to create a hidden layer (or multiple layers) which contains a code to describe the input. There is then a decoder which creates a reconstruction of the input from the hidden layer. An autoencoder can then become useful by having a hidden layer smaller than the input layer, forcing it to create a compressed representation of the data in the hidden layer by learning correlations in the data. This facilitates the classification, visualisation, communication and storage of data [1]. Autoencoders are a form of unsupervised learning, meaning that an autoencoder only needs unlabelled data - a set of input data rather than input-output pairs.

Through an unsupervised learning algorithm, for linear reconstructions the autoencoder will try to learn a function $h_{W,b}(x) \approx x$ $hW,b(x)\approx x$, so as to minimise the mean square difference:

$$L(x,y)=\sum(x-h_{W,b}(x))^2 \quad L(x,y)=\sum(x-hW,b(x))2$$

where x is the input data and y is the reconstruction. However, when the decoder's activation function is the Sigmoid function, the cross-entropy loss function is typically used:

$$L(x,y)=-\sum_{i=1}^{d_x} x_i \log(y_i)+(1-x_i)\log(1-y_i) \quad L(x,y)=-\sum i=1dxxilog(yi)+(1-xi)log(1-yi)$$

We can obtain optimum weights for this by starting with random weights and calculating a gradient. This is done by using the chain rule to back-propagate error derivatives through the decoder network and then the encoder network.

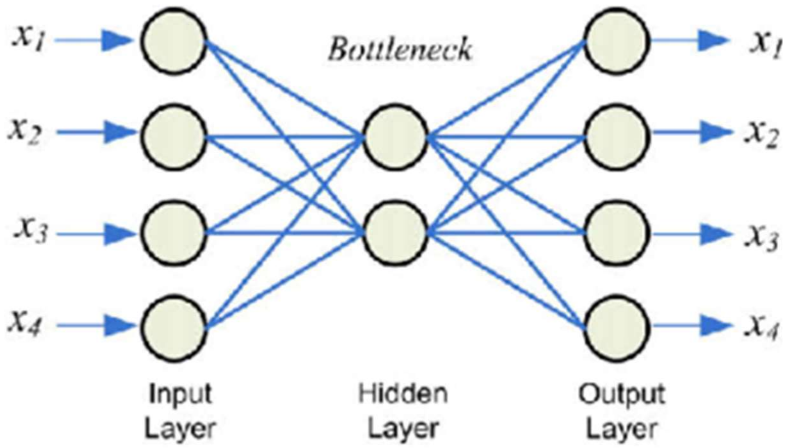

Figure 1: Layers in a autoencoder [2].

2 Uses

The most intuitive application of autoencoders is data compression. Given an 256 x 256px image for example, a representation of a 28 x 28px may be learned, which is easier to handle.

We can also use denoising autoencoders to reconstruct corrupted data, often in the form of images.

Another use is to pre-train deep networks with stacked denoising autoencoders. This is allows us to optimise deep learning solutions and avoid being stuck in local minima as we might be with random initialisation of weights.