Autoencoders

The Basic idea of Autoencoder is based on a fundamental Architecture that allows to replicate data from input to output.

There is symmetry present between the hidden layers

$$\begin{tikzcd} \text{input} \arrow[r] \arrow[rr, bend left, "\text{(equal)}"] & \text{Autoencoder} \arrow[r] & \text{Output} \end{tikzcd}$$

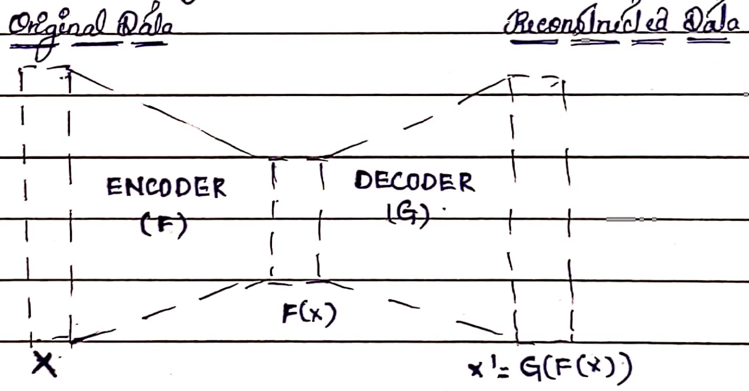
The catch is the output is not exactly same, it is lossy

Input goes through two general stages

(i) Encoder : In this phase, the input is squeezed. A reduction representation of the input is built in here.

The no of units in this layer is the dimensionality of the reduction

(ii) Decoder : Here the input data is reconstructed.



Networks with this kind of architecture can be very powerful and helpful in numerous ways

ex: Unsupervised learning
Matrix factorization
Dimensionality Reduction

Empowering Auto encoders :

The real power of Autoencoders can be harnessed by usage of both

1. Non-linear activation function
2. Deep Architectures

Non Linear Activation function

This can be used both the hidden as well as the output layers.

Non Linear activation function might help the model to map a manifold of an arbitary shape into a reduced representation.

Deep Architectures

Allow the model to create hierarchically reduced representation of the input layer. This increases the representation power of the neural network and can be most natural if one is dealing with domains that can be decomposed hierarchially.

* Autoencoders have the risk of becoming an Identity function, this generally happens when there are more Nodes (or) Neurons in the hidden layer than there are inputs.

DeNoising Autoencoder (DAE)

A Denoising autoencoder is a modification of the original Autoencoder in which instead of giving the original input we give a corrupted or noisy version of input to the Encoder while Decoder loss is calculated concerning Original input only.

Architecture of DAE

It resembles the standard Autoencoder

Encoder

⇒ The encoder is a neural network with one or more hidden layers

⇒ It receives noisy input data instead of the original input and generates an encoding in a low dimensional space

⇒ There are several ways to generate a corrupted input. The most common being adding a Gaussian Noise or randomly masking some of the inputs.

Decoder:

⇒ It is also a hidden layer

⇒ When calculating the loss function it compares the output values with the original input, not with corrupted input.

What does DAE learn?

If DAEs are trained with partially corrupted inputs (eg: masking values), they learn to impute or fill in missing information during the reconstruction process. This makes them useful for tasks involving Incomplete datasets.

If DAEs are trained with partially noisy inputs, they learn to generalize well to unseen, real world data with different levels of noise or corruption as they learn to extract robust features. This is beneficial in various applications where data quality is compromised such as Image denoising or signal processing.

Objective function

Mean square Error (MSE)

If we have input image data in the floating pixel values i.e (0 to 1) or (0 to 255) we use MSE

$$MSE(x,y) = \frac{1}{m}\sum_{1}^{m}(x_i - y_i)^2$$

$$y_i = D(E(x_i * noise))$$

D = decoder

E = encoder

Binary Cross Entropy

If we have input image data in the form of bits pixel i.e either 0 or 1 then we use BCE

$$LL(x,y) = \frac{1}{m}\sum_{1}^{m}(x_i \ln(y_i) + (1-x_i)\ln(1-y_i))$$

- We typically use stochastic gradient design (SGD) or its variants for optimisation.