Twitter Thread by Prashant ■

Prashant ■
@capeandcode

Ever heard of Autoencoders?

The first time I saw a Neural Network with more output neurons than in the hidden layers, I couldn't figure how it would work?!

#DeepLearning #MachineLearning
**Here's a little something about them: ■■**

— Autoencoders

The point of data compression is to convert our input into a smaller representation that we recreate to a larger degree of quality. This smaller representation is what would be passed around and when anyone needed the original, they would reconstruct it from smaller representation, like a zip file.

— Autoencoders are unsupervised neural networks that use machine learning to do this compression for us.

— There are many types of Autoencoders like vanilla autoencoders, deep autoencoders.

Vanilla Autoencoder

The standard, run of the mill autoencoder, is a 2 layer neural network that satisfies
• The hidden layer is smaller size than size of the input and output layer
• The input layer and output layer are the same size.

Input | compressed Representation | output

x_1
x_2
x_3
x_n
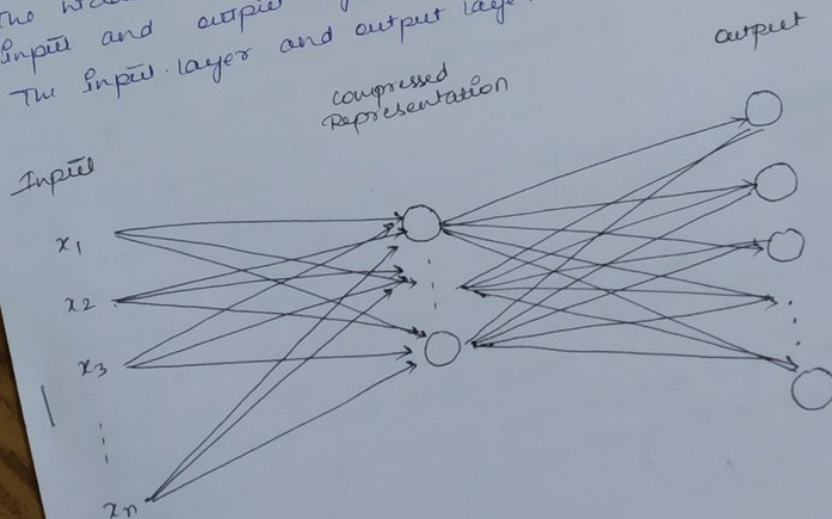
Autoencoders are unsupervised neural networks whose architecture you can picture as two funnels connect from the narrow ends.

These networks are primary focus for compression tasks of data in Machine Learning.

We feed them the data so that they can learn the most important features, a smaller representation while keep the integrity of the data.

Later when someone needs, can just take that small representation and recreate the original, just like a zip file.∎
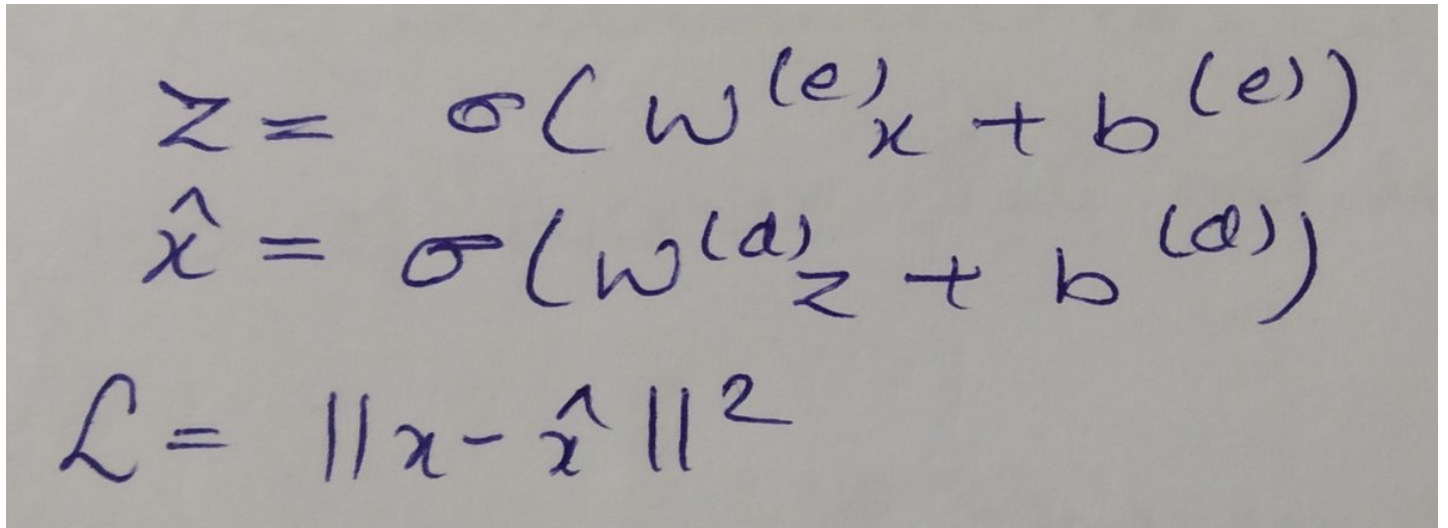
Being unsupervised, they require no labels.
Our inputs and outputs are same and a simple euclidean distance can be used as a loss function for measuring the reconstruction.

Of course, we wouldn't expect a perfect reconstruction.

We can think of an autoencoder having two components, encoder and decoder, represented by the below equations:

We are just trying to minimize the L here. All the backpropagation rules still hold.

$$z = \sigma(w^{(e)}x + b^{(e)})$$
$$\hat{x} = \sigma(w^{(d)}z + b^{(d)})$$
$$\mathcal{L} = \|x - \hat{x}\|^2$$

Advantages over PCA:

■■ Can learn non-linear transformations, with non-linear activation functions and multiple layers.

■■ Doesn't have to learn only from dense layers, can learn from convolutional layers too, better for images, videos right?

■■ More efficient to learn several layers with auto-encoders rather than one huge transformation with PCA

■■ Can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder /decoder

Some Common Applications:

■ Image Colouring
■ Feature Variation
■ Dimensionality Reduction
■ Denoising Image
■ Watermark Removal

Some famous types of autoencoders:

■ Convolution Autoencoders
■ Sparse Autoencoders
■ Deep Autoencoders
■ Contractive Autoencoders

Here's the first implementation that I did for dimensionality reduction a couple years, minimal code.
■https://t.co/AfAdbA6zMi