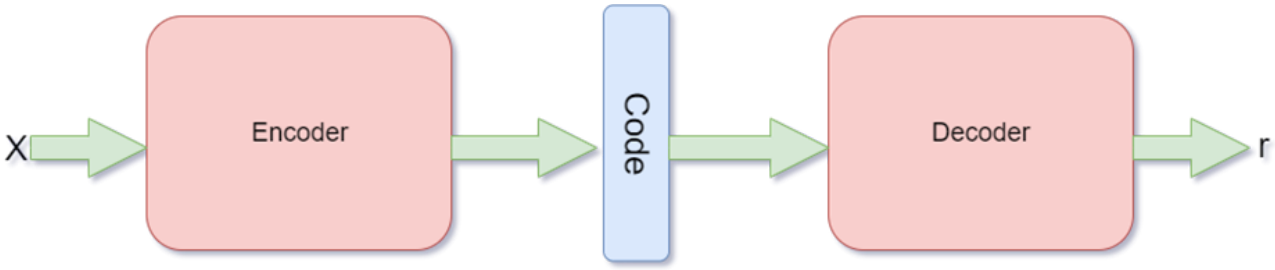
DL M5 (p-1)

techworldthink • March 22, 2022

Autoencoders

- Autoencoders are an unsupervised learning technique that we can use to learn efficient data encodings
- Basically, autoencoders can learn to map input data to the output data
- While doing so, they learn to encode the data. And the output is the compressed representation of the input data
- The main aim while training an autoencoder neural network is dimensionality reduction
- *Autoencoding" is a data compression algorithm where the compression and decompression functions are 1) data-specific, 2) lossy, and 3) learned automatically from examples rather than engineered by a human*

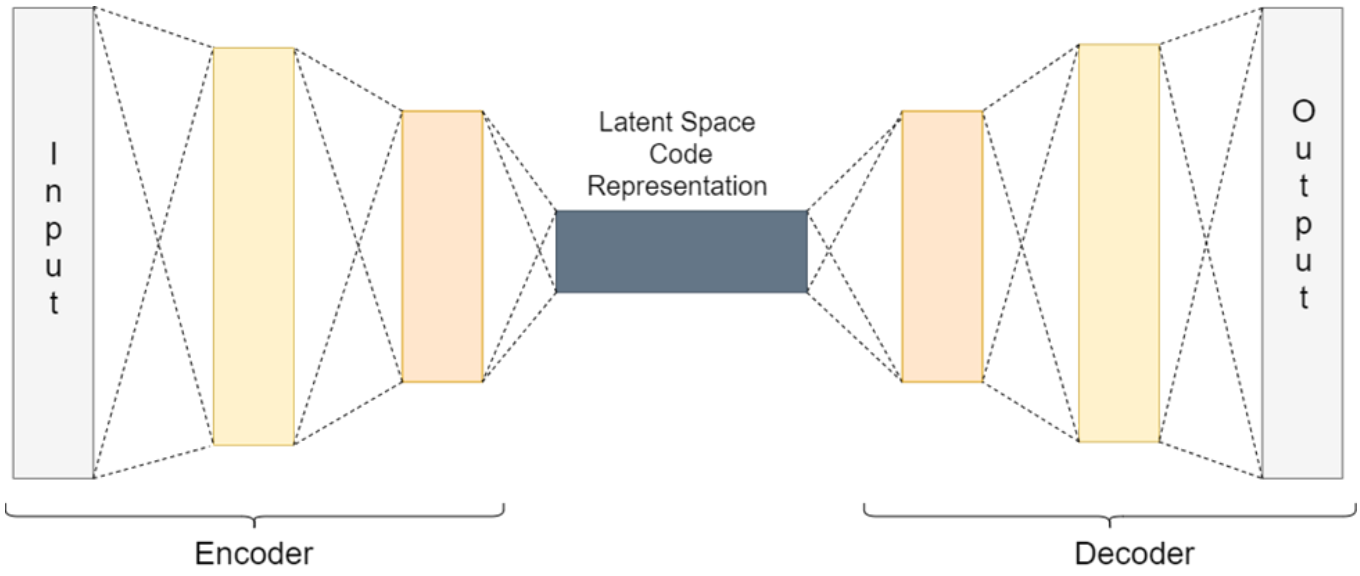


- An autoencoder is a feed-forward neural net whose job it is to take an input x and predict x
- In another words, autoencoders are neural networks that are trained to copy their inputs to their outputs
- It consists of Encoder $h = f(x)$ Decoder $r = g(h)$
- An autoencoder is a data compression algorithm
- A hidden layer describes the code used to represent the input

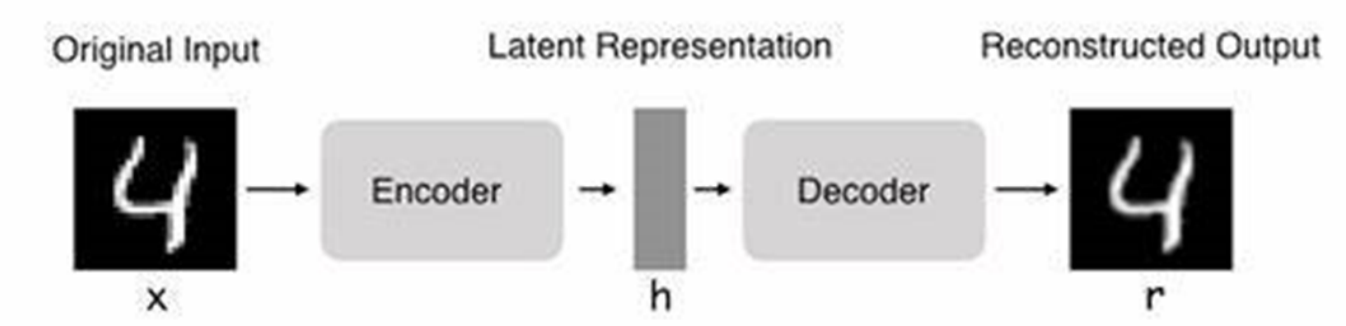
- It maps input to output through a compressed representation code

The Principle Behind Autoencoder

- In an autoencoder, there are two parts, an ***encoder,*** and a ***decoder***
- First, the encoder takes the input and encodes it
- For example, let the input data be x. Then, we can define the encoded function as f(x)
- Between the encoder and the decoder, there is also an ***internal hidden layer***
- Let's call this hidden layer h. This hidden layer learns the coding of the input that is defined by the encoder
- So, basically after the encoding, we get h = f(x)
- Finally, the decoder function tries to reconstruct the input data from the hidden layer coding. If we consider the decoder function as g, · then the reconstruction can be defined as, r=g(f(x)) r=g(h)



The **latent space** is simply **a representation of compressed data in which similar data points** are closer together in space. Latent space is useful for learning data features and for finding simpler representations of data for analysis



An autoencoder should be able to reconstruct the input data efficiently but by learning the useful properties rather than memorizing it

1) Autoencoders are data-specific, which means that they will only be able to compress data similar to what they have been trained on. This is different from, say, the MPEG-2 Audio Layer III (MP3) compression algorithm, which only holds assumptions about "sound" in general, but not about specific types of sounds. An autoencoder trained on pictures of faces would do a rather poor job of compressing pictures of trees, because the features it would learn would be face-specific.

2) Autoencoders are lossy, which means that the decompressed outputs will be degraded compared to the original inputs (similar to MP3 or JPEG compression). This differs from lossless arithmetic compression.

3) Autoencoders are learned automatically from data examples, which is a useful property: it means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input. It doesn't require any new engineering, just appropriate training data.

Autoencoder architectures

1) Undercomplete autoencoder

- The simplest architecture for constructing an autoencoder is to constrain the number of nodes present in the hidden layer(s) of the network, limiting the amount of information that can flow through the network.
- By penalizing the network according to the reconstruction error, our model can learn the most important attributes of the input data and how to best reconstruct the original input from an "encoded" state.

- Ideally, this encoding will learn and describe latent attributes of the input data.
- An undercomplete autoencoder has no explicit regularization term - we simply train our model according to the reconstruction loss.
- we discussed that we want our autoencoder to learn the important features of the input data.It should do that instead of trying to memorize and copy the input data to the output data
- We can do that if we make the hidden coding data to have less dimensionality than the input data
- In an autoencoder, when the encoding h has a smaller dimension than x, then it is called an undercomplete autoencoder
- The above way of obtaining reduced dimensionality data is the same as PCA. In PCA also, we try to try to reduce the dimensionality of the original data
- The loss function for the above process can be described as, $L(x,r)=L(x,g(f(x)))$

2) Regularized Autoencoders

- In undercomplete autoencoders, we have the coding dimension to be less than the input dimension.
- We also have ***overcomplete*** autoencoder in which the coding dimension is the same as the input dimension. But this again raises the issue of the model not learning any useful features and simply copying the input.
- One solution to the above problem is the use of ***regularized autoencoder***. When training a regularized autoencoder we need not make it undercomplete. We can choose the coding dimension and the capacity for the encoder and decoder according to the task at hand
- To properly train a regularized autoencoder, we choose loss functions that help the model to learn better and capture all the essential features of the input data.
- Two common ways of implementing regularized autoencoders are **sparse Autoencoders , Denoising Autoencoders**

2.1 sparse Autoencoders

- In ***sparse autoencoders,*** we use a loss function as well as an additional penalty for sparsity. Specifically, we can define the loss function as, $L(x,g(f(x))) + \Omega(h)$. where $\Omega(h)$ is the additional sparsity penalty on the code h.
- Adding a penalty such as the sparsity penalty helps the autoencoder to capture many of the useful features of data and not simply copy it.

2.2 Denoising Autoencoders

- In sparse autoencoders, we have seen how the loss function has an additional penalty for the proper coding of the input data.
- *But what if we want to achieve similar results without adding the penalty?* In that case, we can use something known as ***denoising autoencoder***
- We can change the reconstruction procedure of the decoder to achieve that. Until now we have seen the decoder reconstruction procedure as $r(h) = g(f(x))$ and the loss function as $L(x,g(f(x)))$.
- Now, consider adding noise to the input data to make it $x\sim$ instead of x. Then the loss function becomes, $L(x,g(f(x\sim)))$
- For a proper learning procedure, now the autoencoder will have to minimize the above loss function. And to do that, it first will have to cancel out the noise, and then perform the decoding.
- In a denoising autoencoder, the model cannot just copy the input to the output as that would result in a noisy output. While we update the input data with added noise, we can also use overcomplete autoencoders without facing any problems.

3) Variational Autoencoders (VAEs)

- VAEs are a type of generative model like GANs (Generative Adversarial Networks)
- A generative adversarial network, or GAN, is a deep neural network framework which is able to learn from a set of training data and generate new data with the

same characteristics as the training data

- For example, a generative adversarial network trained on photographs of human faces can generate realistic-looking faces which are entirely fictitious
- Generative adversarial networks consist of **two neural networks**, the **generator and the discriminator,** which compete against each other. The generator is trained to produce fake data, and the discriminator is trained to distinguish the generator's fake data from real examples. If the generator produces fake data that the discriminator can easily recognize as implausible, such as an image that is clearly not a face, the generator is penalized. Over time, the generator learns to generate more plausible examples
- Like other autoencoders, variational autoencoders also consist of an encoder and a decoder. But here, the decoder is the generator model. Variational autoencoders also carry out the reconstruction process from the latent code space. But in VAEs, the latent coding space is continuous

Applications of Autoencoders

1) Dimensionality Reduction and PCA

- When we use undercomplete autoencoders, we obtain the latent code space whose *dimension is less than the input*.
- Dimensionality reduction refers to techniques for **reducing the number of input variables in training data**. When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data.

2) Image Denoising and Image Compression

- Denoising autoencoder can be used for the purposes of image denoising. Autoencoders are able to cancel out the noise in images before learning the important features and reconstructing the images.

- Autoencoder can also be used for image compression to some extent. More on this in the limitations part.

3) Information Retrieval

- When using deep autoencoders, then reducing the dimensionality is a common approach. This reduction in dimensionality leads the encoder network to capture some really important information

Limitations of Autoencoders

We have seen how autoencoders can be used for image compression and reconstruction of images. But in reality, they are not very efficient in the process of compressing images. Also, they are only efficient when reconstructing images similar to what they have been trained on.

Due to the above reasons, the practical usages of autoencoders are limited.