What are the main tasks that autoencoders are used for?

Ans. Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.

Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data.

Autoencoders used for :-

1.Dimensionality Reduction

2.Image Compression

3.Image Denoising

4.Feature Extraction

5.Image generation

6.Sequence to sequence prediction

7.Recommendation system

Suppose you want to train a classifier, and you have plenty of unlabeled training data but only a few thousand labeled instances. How can autoencoders help? How would you proceed?

Ans. Rather than limiting the model capacity by keeping the encoder and decoder shallow and the code size small, regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output.

If an autoencoder perfectly reconstructs the inputs, is it necessarily a good autoencoder? How can you evaluate the performance of an autoencoder?

Ans. If you consider conventional autoencoder function, yes, it is a good autoencoder. In practice, efficiency of autoencoder depends on how well it reconstructs and also on how robust it is to noise in different scenes.Common practice is to add noise sampled from input distribution to the input space to make sure autoencoder, vanilla or VAE, learns to reconstruct the input more robustly regardless of scenic distortions.However, maybe your goal never was reconstruction and thus it doesn’t matter how good reconstruction is. Maybe you wanted to learn features and leverage it for other use. In that case, you wouldn’t care, mostly, about how well reconstruction happens. It is known that noise in input space doesn’t necessary help in better converage of feature space and thus feature learning is hampered. So, community came up with idea of introducing noise in the feature space instead of input space. It will obviously hurt the reconstruction but definitely learned features would be better and overall your feature vector would be more definitive of the latent space as a whole.

What are undercomplete and overcomplete autoencoders? What is the main risk of an excessively undercomplete autoencoder? What about the main risk of an overcomplete autoencoder?

Ans. The answer provided below has been developed in a clear step by step manner.

Step: 1

AutoEncoders - Learning can be supervised as well as unsupervised.

Autoencoder is basically form of unsupervised learing.

Autoencoder can be Undercomplete or Overcomplete.

Undercomplete - In Undercomplete the dimension of encoding output is less than dimension of input.The only difference between Undercomplete and Overcomplete is size of dimension of encoding output.

Overcomplete - In Overcomplete the dimension of encoding output is bigger than dimension of input.The only difference between Undercomplete and Overcomplete is size of dimension of encoding output.

Excessive means much larger than required.

In Undercomplete autoencoder the dimension of encoding output is less than dimension of input.

So their will be a risk that Undercomplete autoencoder may not be able to make inputs.

In Case of Overcomplete the dimension of encoding output is bigger than dimension of input.

So their may be case that it may just do copy of input to output without any useful learning/feature.

Explanation:Please refer to solution in this step.

Answer:

AutoEncoders - Learning can be supervised as well as unsupervised.

Autoencoder is basically form of unsupervised learing.

Autoencoder can be Undercomplete or Overcomplete.

Undercomplete - In Undercomplete the dimension of encoding output is less than dimension of input.

The only difference between Undercomplete and Overcomplete is size of dimension of encoding output.

Overcomplete - In Overcomplete the dimension of encoding output is bigger than dimension of input.

The only difference between Undercomplete and Overcomplete is size of dimension of encoding output.

Excessive means much larger than required.

In Undercomplete autoencoder the dimension of encoding output is less than dimension of input.

So their will be a risk that Undercomplete autoencoder may not be able to make inputs.

In Case of Overcomplete the dimension of encoding output is bigger than dimension of input.

So their may be case that it may just do copy of input to output without any useful learning/feature.

How do you tie weights in a stacked autoencoder? What is the point of doing so?

Ans. Tie wieghts in a stack encoder

When we are dealing with a symmetrical autoencoder, a good practice is to tie the weights of the decoder layers to the weights of the encoder layers. With this technique we halve the number of weights in our model, speeding training and limiting the risk of overfitting, since we don't have to learn the weights of the decoder anymore, we just learn the weights of the encoder and set the weights of the decoder accordingly. What I don't understand is the value assigned to the weights of the decoder. Let's say that the autoencoder has a total of NN layers (without counting the input layer), so layer 11 is the first hidden layer, layer N/2N/2 is the codding layer, and layer NN is the output layer. Let's also say that WLWL represents the connection weights of the LthLth layer.From what I've read so far, if we tie the weights of the decoder layers to the weights of the encoder layer, then the weights of the decoder layers will be:

WN−L+1=W⊺LWN−L+1=WL⊺

where ⊺⊺ denotes the transpose and LL ranges from 1,2,...,N/21,2,...,N/2.

Why are we using the transpose of the encoder layers weights as the decoder layer weights? Since the encoder has the job of projecting our data into a lower dimension, and then the decoder maps this projection back to the original representation of the data, wouldn't it make more sense to have the weights of the decoder be the inverse of the encoder weights, not the transpose? (Or at least the pseudo-inverse)If we would use the inverse, that weight matrix would try to project the data back to its original space, it would try to undo the initial projection, and therefore it would try to recreate the initial input, which is what the autoencoder is trying to achieve. But we're not using the inverse. We're using the transpose. Autoencoders with tied weights have some important advantages :

It's easier to learn.

In linear case it's equvialent to PCA - this may lead to more geometrically adequate coding.

Tied weights are sort of regularisation.

What is a generative model? Can you name a type of generative autoencoder?

Ans. A generative model includes the distribution of the data itself, and tells you how likely a given example is. For example, models that predict the next word in a sequence are typically generative models (usually much simpler than GANs) because they can assign a probability to a sequence of words. An example of a generative model might be one that is trained on collections of images from the real world in order to generate similar images. The model might take observations from a 200GB set of images and reduce them into 100MB of weights. Weights can be thought of as reinforced neural connections.

What is a GAN? Can you name a few tasks where GANs can shine?

Ans. GAN-Generative adversarial networks are a revenyand exciting innovation in machine learning.these are generative models which creates new data that resemble your past data.it is un- supervised and use a cooperative zero sum game framework to learn.

These are used for high fidelity natural image synthesis,data augmentation tasks,improving image compressions.

What are the main difficulties when training GANs?

Ans. Mode collapse is one of the hardest problems to solve in GAN. A complete collapse is not common but a partial collapse happens often. The images below with the same underlined color look similar and the mode starts collapsing.