Autoencoders

Autoencoders are a special type of neural networks fully-connected that work by trying to reproduce the data input at the output of the network. Although this process may sound "trivial" and even useless, the architecture of these networks allows them to perform some very interesting tasks that differentiate them from the neural networks that we have seen so far.

Specifically, three of its main applications are:

- Dimensionality reduction. Networks are trained with unlabeled (unsupervised) data sets to learn an efficient and reduced representation of the input data, which allows reduce the dimension of the input data.
- Neural network pretraining. The autoencoders can facilitate the detection of the most relevant attributes, and can be used for pre-training unsupervised neural networks.
- Finally, they can also be used for the generation of new synthetic data that allow increasing the training data sets. That is, they can be used to generate "like" data that they receive as input, in such a way that they can later be use for training other neural networks (or other machine learning algorithms).

Basic structure

An autoencoder always has two clearly differentiated parts:

- Encoder, which is responsible for converting the inputs to an internal representation (latent space), usually smaller in dimension than the input data (latent dimension).
- Decoder, which is responsible for transforming the internal representation at the output of the network.

Figure 1 shows the basic structure of an autoencoder. Although it may seem similar to the structure of a network like the ones we have seen before (both are feed forward and fully-connected), has some relevant differences. First of all, the autoencoders have the same number of neurons in the output layer than in the input layer, that it will try to reproduce in the output the input that it has received. Second, the hidden layer (or hidden layers) must have a lower number of neurons than the input and output layers, since otherwise the task of reproducing the input in the output would be trivial. Therefore, the representation internal must preserve the input information in a lower dimensional format.

The concepts seen above about the initialization of parameters, activation functions, regularization, etc., also are applicable in the case of autoencoders. The main difference is that in

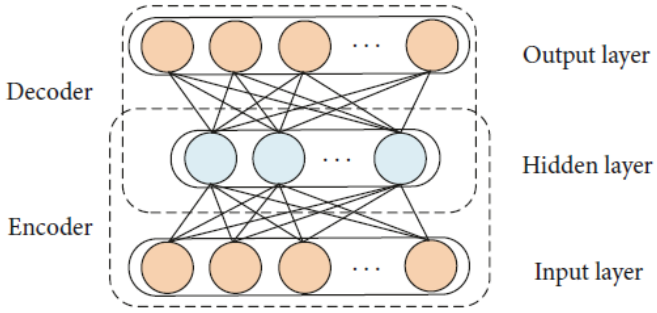

Figure 1: Basic structure of autoencoder

this case the target class or value is not used of the training set. The desired output (and that we will use to calculate the error that the network commits) will be the same input. For this reason, they are not usually used softmax neurons in the output layer.

Pre-training with autoencoders

Pretraining with autoencoders is a technique that allows to reduce the dimensionality of the data before training of the model that will be in charge of the classification task or regression.

Although this technique has traditionally been applied to reduce training time (similar to using of dimensionality reduction algorithms), advances and improvements in the calculation capacity have caused that currently it is not used much to improve the speed of the training processes. Even so, this technique continues being very popular when we have a dataset with few samples (n) in relation to the number of attributes (p) and allows to improve the ratio between instances and attributes.

The idea of this scheme, as shown in Fig. 2, it is quite simple, but also very powerful. First of all, the autoencoder must be trained, either with one or more hidden layers, using all available data (whether labeled or not).

Once the autoencoder training is finished, they are copied the weight and bias values of the initial layers up to lower dimensional layer. At the output of the coding layer, the neural network is built (usually a fully connected network, although it could be another machine learning model). The output of this second network will be the target classes or values of the problem to be solved and therefore the training will be done in a similar way as we have seen before. It is important to highlight that during the training of this second network, the values obtained (weights and bias) from the autoencoder are usually kept frozen.

This pre-training process can be very useful in different contexts, such as:

- When a small set of labeled data is available with high dimensionality. By reducing the number of attributes with which the neural network works, we facilitate the learning task.
- When you have a large data set, but in which only part is labeled. In these cases

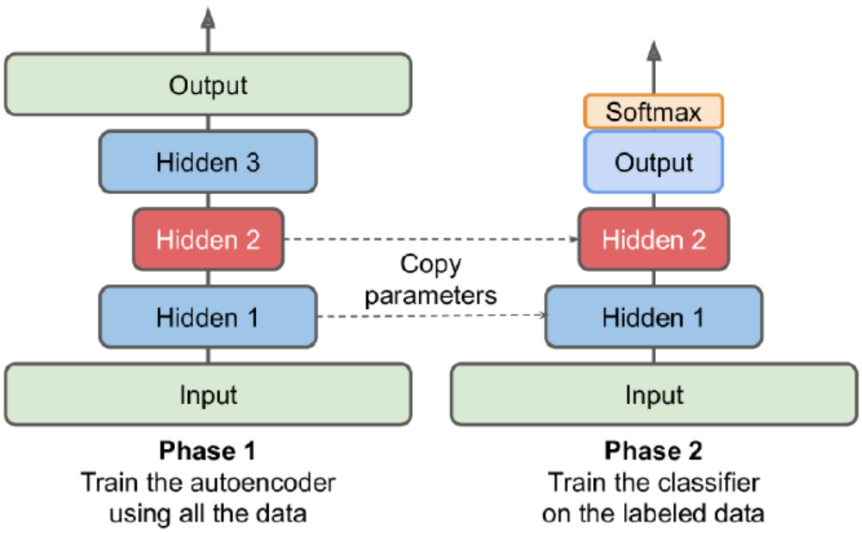

Figure 2: Pre-training

it is possible to use the entire data set to train the autoencoder (and get a good representation of the data) and then train the model using only available labeled data.

Stacked Autoencoders (SAE)

The structure of SAEs (Figure 3) is stacking n autoencoders into n hidden layers by an unsupervised layerwise learning algorithm and then **fine-tuned** by a supervised method. So the SAEs based method can be divided into three steps:

1- Train the first autoencoder by input data and obtain the learned feature vector;

2- The feature vector of the former layer is used as the input for the next layer, and this procedure is repeated until the training completes.

3- After all the hidden layers are trained, backpropagation algorithm is used to minimize the cost function and update the weights with labeled training set to achieve fine tuning.

Denoising Autoencoder

A Denoising Autoencoder is a modification on the autoencoder to prevent the network learning the identity function. Specifically, if the autoencoder is too big, then it can just learn the data, so the output equals the input, and does not perform any useful representation learning or dimensionality reduction. Denoising autoencoders solve this problem by corrupting the input data on purpose, adding noise or masking some of the input values.

When calculating the Loss function, it is important to compare the output values with the original input, not with the corrupted input. That way, the risk of learning the identity function instead of extracting features is eliminated.

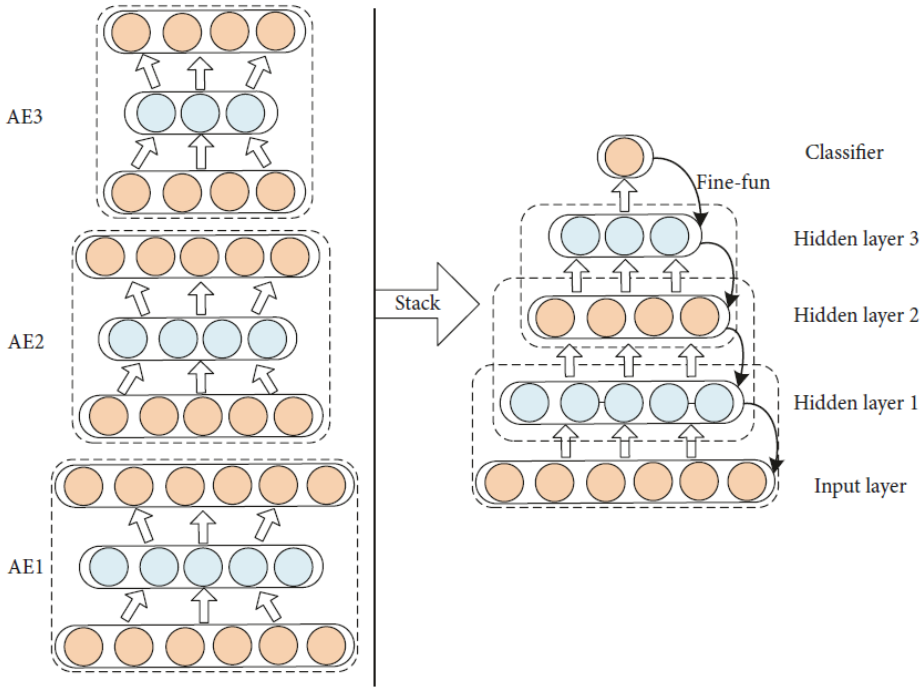

Figure 3: Structure of stacked autoencoder

Variational Autoencoders

Standard autoencoders learn to generate compact representations and reconstruct their inputs well, but asides from a few applications like denoising autoencoders, they are fairly limited. The fundamental problem with autoencoders, for generation, is that the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation.

Variational Autoencoders (VAEs) have one fundamentally unique property that separates them from vanilla autoencoders, and it is this property that makes them so useful for generative modeling: their latent spaces are, by design, continuous, allowing easy random sampling and interpolation.

We will see this topic in more detail in the last unit of the course.