PCA vs auto-encoder

An auto-encoder is typically considered and depicted as an encoder-decoder structure shown

in Figure 2. As a first task, **consider an auto-encoder that mimics linear PCA**. Explain how

you can do this and what the differences are.

In terms of a neural network the auto-encoder will contain an input layer, one hidden layer and an output layer. It is optional to practically experiment, but we ask you to elaborate theoretically on the **expected differences in results when PCA is framed into a neural network**. We do not expect any mathematical proves, but rather want you to show us your understandings.



Figure 2: Typical visualization of auto-encoders or encoder-decoder networks (source).

The autoencoder tries to learn a function hW;b(x) ≈ x. In other words, it is trying to learn an approximation to the identity function, so as to output ^x that is similar to x. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units (called a sparse autoencoder), we can discover interesting structure about the data. In fact, this simple autoencoder often ends up learning a low-dimensional representation similar to PCAs.

Both PCA and autoencoder can do demension reduction

PCA is restricted to a linear map, while auto encoders can have nonlinear enoder/decoders.

A single layer auto encoder with linear transfer function is nearly equivalent to PCA, where nearly means that the W found by AE and PCA won't be the same--but the subspace spanned by the respective W's will.

One thing to note is that the hidden layer in an AE can be of greater dimensionality than that of the input. In such cases AE's may not be doing dimensionality reduction. In this case we perceive them as doing a transformation from one feature space to another wherein the data in the new feature space disentangles factors of variation.

Reading [Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima](http://www.sciencedirect.com/science/article/pii/0893608089900142) where the proof is given: ''In the auto-associative case ... and therefore the unique locally and globally optimal map W is the orthogonal projection onto the space spanned by the first pp eigenvectors of ΣXX''

This then exactly the correspondent space as spanned by PCA. The paper uses a linear autoencoder, i.e., no non-linear activation function. That is why its weights span the same subspace spanned by PCA exactly.

The weights of a linear autoencoder span the same subspace as the principal components found by PCA, but they are not the same vectors. In particular, they are not an orthogonal basis. This is true, **however** we can easily recover the principal components loading vectors from the autoencoder weights. More elaborate answer at [[1]](#footnote-1)

A lot have been written about using a linearly activated autoencoder (AE) to approximate principal component analysis (PCA). From a math point of view, minimizing the reconstruction error in PCA is the same as AE [3].

Encoder part will be equivalent to PCA if linear encoder, linear decoder, square error loss function with normalized inputs are used. Which means PCA is restricted to linear maps only whereas autoencoders are not.

However, why limit ourselves to linear transformations? Neural nets are very flexible, therefore we can introduce non-linearities by using non-linear activation functions [4]. Additionally, with an increasing amount of features, PCA will result in slower processing compared with an AE. Our Hypothesis is that the subspace spanned by the AE will be similar to the one found by PCA [5].[[2]](#footnote-2)

An autoencoder doesn’t have to learn dense (affine) layers; it can use convolutional layers to learn too, which could be better for video, image and series data. It may be more efficient, in terms of model parameters, to learn several layers with an autoencoder rather than learn one huge transformation with PCA. An autoencoder gives a representation as the output of each layer, and maybe having multiple representations of different dimensions is useful. An autoencoder could let you make use of pretrained layers from another model, to apply transfer learning to prime the encoder/decoder.

PCA can work with very little data, an autoencoder can overfit if you have not enough data and that's why autoencoders are really sparse autoencoders.

1. <https://stats.stackexchange.com/questions/120080/whatre-the-differences-between-pca-and-autoencoder> [↑](#footnote-ref-1)
2. <https://towardsdatascience.com/pca-vs-autoencoders-1ba08362f450> [↑](#footnote-ref-2)