1. Non-negative matrix factorization (NMF)

2. Non-negative matrix factorization

NMF stands for "non-negative matrix factorization". NMF, like PCA, is a dimension reduction technique. In constract to PCA, however, NMF models are interpretable. This means an NMF models are easier to understand yourself, and much easier for you to explain to others. NMF can not be applied to every dataset, however. It is required that the sample features be "non-negative", so greater than or equal to 0.

3. Interpretable parts

NMF achieves its interpretability by decomposing samples as sums of their parts. For example, NMF decomposes documents as combinations of common themes,

4. Interpretable parts

and images as combinations of common patterns. You'll learn about both these examples in detail later. For now, let's focus on getting started.

5. Using scikit-learn NMF

NMF is available in scikit learn, and follows the same fit/transform pattern as PCA. However, unlike PCA, the desired number of components must always be specified. NMF works both with numpy arrays and sparse arrays in the csr_matrix format.

6. Example word-frequency array

Let's see an application of NMF to a toy example of a word-frequency array. In this toy dataset, there are only 4 words in the vocabulary, and these correspond to the four columns of the word-frequency array. Each row represents a document, and the entries of the array measure the frequency of each word in the document using what's known as "tf-idf". "tf" is the frequency of the word in the document. So if 10% of the words in the document are "datacamp", then the tf of "datacamp" for that document is point-1. "idf" is a weighting scheme that reduces the influence of frequent words like "the".

7. Example usage of NMF

Let's now see how to use NMF in Python. Firstly, import NMF. Create a model, specifying the desired number of components. Let's specify 2. Fit the model to the samples, then use the fit model to perform the transformation.

8. NMF components

Just as PCA has principal components, NMF has components which it learns from the samples, and as with PCA, the dimension of the components is the same as the dimension of the samples. In our example, for instance, there are 2 components, and they live in 4 dimensional space, corresponding to the 4 words in the vocabulary. The entries of the NMF components are always non-negative.

9. NMF features

The NMF feature values are non-negative, as well. As we saw with PCA, our transformed data in this example will have two columns, corresponding to our two new features. The features and the components of an NMF model can be combined to approximately reconstruct the original data samples.

10. Reconstruction of a sample

Let's see how this works with a single data sample. Here is a sample representing a document from our toy dataset, and here are its NMF feature values. Now if we multiply each NMF components by the corresponding NMF feature value, and add up each column, we get something very close to the original sample.

11. Sample reconstruction

So a sample can be reconstructed by multiplying the NMF components by the NMF feature values of the sample, and adding up. This calculation also can be expressed as what is known as a product of matrices. We won't be using that point of view, but that's where the "matrix factorization", or "MF", in NMF comes from.

12. NMF fits to non-negative data only

Finally, remember that NMF can only be applied to arrays of non-negative data, such as word-frequency arrays. In the next video, you'll construct another example by encoding collections of images as non-negative arrays. There are many other great examples as well, such as arrays encoding audio spectrograms, and arrays representing the purchase histories on e-Commerce sites.