**Non-negative matrix factorization (NMF)**

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.

**NMF learns interpretable parts**

In this video, you'll learn that the components of NMF represent patterns that frequently occur in the samples.

**2. Example: NMF learns interpretable parts**

00:00 - 00:00

Let's consider a concrete example, where scientific articles are represented by their word frequencies. There are 20000 articles, and 800 words. So the array has 800 columns.

**3. Applying NMF to the articles**

00:00 - 00:00

Let's fit an NMF model with 10 components to the articles. The 10 components are stored as the 10 rows of a 2-dimensional numpy array.

**4. NMF components are topics**

00:00 - 00:00

The rows, or components, live in an 800-dimensional space - there is one dimension for each of the words. Aligning the words of our vocabulary with the columns of the NMF components allows them to be interpreted.

**5. NMF components are topics**

00:00 - 00:00

Choosing a component, such as this one, and looking at which words have the highest values,

**6. NMF components are topics**

00:00 - 00:00

we see that they fit a theme: the words are 'species', 'plant', 'plants', 'genetic', 'evolution' and 'life'.

**7. NMF components are topics**

00:00 - 00:00

The same happens if any other component is considered.

**8. NMF components**

00:00 - 00:00

So if NMF is applied to documents, then the components correspond to topics, and the NMF features reconstruct the documents from the topics. If NMF is applied to a collection of images, on the other hand, then the NMF components represent patterns that frequently occur in the images. In this example, for instance, NMF decomposes images from an LCD display into the individual cells of the display. This example you'll investigate for yourself in the exercises. To do this, you'll need to know how to represent a collection of images as a non-negative array.

**9. Grayscale images**

00:00 - 00:00

An image in which all the pixels are shades of gray ranging from black to white is called a "grayscale image". Since there are only shades of grey, a grayscale image can be encoded by the brightness of every pixel. Representing the brightness as a number between 0 and 1, where 0 is totally black and 1 is totally white, the image can be represented as 2-dimensional array of numbers.

**10. Grayscale image example**

00:00 - 00:00

Here, for example, is a grayscale photo of the moon!

**11. Grayscale images as flat arrays**

00:00 - 00:00

These 2-dimensional arrays of numbers can then be flattened by enumerating the entries. For instance, we could read-off the values row-by-row, from left-to-right and top to bottom.

**12. Grayscale images as flat arrays**

00:00 - 00:00

The grayscale image is now represented by a flat array of non-negative numbers.

**13. Encoding a collection of images**

00:00 - 00:00

A collection of grayscale images of the same size can thus be encoded as a 2-dimensional array, in which each row represents an image as a flattened array, and each column represents a pixel. Viewing the images as samples, and the pixels as features, we see that the data is arranged similarly to the word frequency array. Indeed, the entries of this array are non-negative, so NMF can be used to learn the parts of the images.

**14. Visualizing samples**

00:00 - 00:00

It's difficult to visualize an image by just looking at the flattened array. To recover the image, use the reshape method of the sample, specifying the dimensions of the original image as a tuple. This yields the 2-dimensional array of pixel brightnesses. To display the corresponding image, import pyplot, and pass the 2-dimensional array to the plt dot imshow function.

**Building recommender systems using NMF**

**2. Finding similar articles**

00:01 - 00:21

Suppose that you are an engineer at a large online newspaper. You've been given the task of recommending articles that are similar to the article currently being read by a customer. Given an article, how can you find articles that have similar topics? In this video, you'll learn how to solve this problem, and others like it, by using NMF.

**3. Strategy**

00:21 - 00:48

Our strategy for solving this problem is to apply NMF to the word-frequency array of the articles, and to use the resulting NMF features. You learned in the previous videos these NMF features describe the topic mixture of an article. So similar articles will have similar NMF features. But how can two articles be compared using their NMF features? Before answering this question, let's set the scene by doing the first step.

**4. Apply NMF to the word-frequency array**

00:48 - 01:08

You are given a word frequency array articles corresponding to the collection of newspaper articles in question. Import NMF, create the model, and use the fit\_transform method to obtain the transformed articles. Now we've got NMF features for every article, given by the columns of the new array.

**5. Strategy**

01:08 - 01:13

Now we need to define how to compare articles using their NMF features.

**6. Versions of articles**

01:13 - 01:27

Similar documents have similar topics, but it isn't always the case that the NMF feature values are exactly the same. For instance, one version of a document might use very direct language,

**7. Versions of articles**

01:27 - 01:40

whereas other versions might interleave the same content with meaningless chatter. Meaningless chatter reduces the frequency of the topic words overall, which reduces the values of the NMF features representing the topics.

**8. Versions of articles**

01:40 - 01:48

However, on a scatter plot of the NMF features, all these versions lie on a single line passing through the origin.

**9. Cosine similarity**

01:48 - 02:12

For this reason, when comparing two documents, it's a good idea to compare these lines. We'll compare them using what is known as the cosine similarity, which uses the angle between the two lines. Higher values indicate greater similarity. The technical definition of the cosine similarity is out the scope of this course, but we've already gained an intuition.

**10. Calculating the cosine similarities**

02:12 - 02:34

Let's see now how to compute the cosine similarity. Firstly, import the normalize function, and apply it to the array of all NMF features. Now select the row corresponding to the current article, and pass it to the dot method of the array of all normalized features. This results in the cosine similarities.

**11. DataFrames and labels**

02:34 - 03:03

With the help of a pandas DataFrame, we can label the similarities with the article titles. Start by importing pandas. After normalizing the NMF features, create a DataFrame whose rows are the normalized features, using the titles as an index. Now use the loc method of the DataFrame to select the normalized feature values for the current article, using its title 'Dog bites man'. Calculate the cosine similarities using the dot method of the DataFrame.

**12. DataFrames and labels**

03:03 - 03:17

Finally, use the nlargest method of the resulting pandas Series to find the articles with the highest cosine similarity. We see that all of them are concerned with 'domestic animals' and/or 'danger'!