**Visualizing the PCA transformation**

In the next two chapters you'll learn techniques for dimension reduction.

**Dimension reduction**

Dimension reduction finds patterns in data, and uses these patterns to re-express it in a compressed form. This makes subsequent computation with the data much more efficient, and this can be a big deal in a world of big datasets. However, the most important function of dimension reduction is to reduce a dataset to its "bare bones", discarding noisy features that cause big problems for supervised learning tasks like regression and classification. In many real-world applications, it's dimension reduction that makes prediction possible.

**Principal Component Analysis**

In this chapter, you'll learn about the most fundamental of dimension reduction techniques. It's called "Principal Component Analysis", or "PCA" for short. PCA performs dimension reduction in two steps, and the first one, called "de-correlation", doesn't change the dimension of the data at all. It's this first step that we'll focus on in this video.

**PCA aligns data with axes**

In this first step, PCA rotates the samples so that they are aligned with the coordinate axes. In fact, it does more than this: PCA also shifts the samples so that they have mean zero. These scatter plots show the effect of PCA applied to two features of the wine dataset. Notice that no information is lost - this is true no matter how many features your dataset has. You'll practice visualizing this transformation in the exercises.

**PCA follows the fit/transform pattern**

scikit-learn has an implementation of PCA, and it has fit and transform methods just like StandardScaler. The fit method learns how to shift and how to rotate the samples, but doesn't actually change them. The transform method, on the other hand, applies the transformation that fit learned. In particular, the transform method can be applied to new, unseen samples.

**Using scikit-learn PCA**

Let's see PCA in action on the some features of the wine dataset. Firstly, import PCA. Now create a PCA object, and fit it to the samples. Then use the fit PCA object to transform the samples. This returns a new array of transformed samples.

**PCA features**

This new array has the same number of rows and columns as the original sample array. In particular, there is one row for each transformed sample. The columns of the new array correspond to "PCA features", just as the original features corresponded to columns of the original array.

**PCA features are not correlated**

It is often the case that the features of a dataset are correlated. This is the case with many of the features of the wine dataset, for instance. However, PCA, due to the rotation it performs, "de-correlates" the data, in the sense that the columns of the transformed array are not linearly correlated.

**Pearson correlation**

Linear correlation can be measured with the Pearson correlation. It takes values between -1 and 1, where larger values indicate a stronger correlation, and 0 indicates no linear correlation. Here are some examples of features with varying degrees of correlation.

**Principal components**

Finally, PCA is called "principal component analysis" because it learns the "principal components" of the data. These are the directions in which the samples vary the most, depicted here in red. It is the principal components that PCA aligns with the coordinate axes.

After a PCA model has been fit, the principal components are available as the components attribute. This is numpy array with one row for each principal component.

**Intrinsic dimension**

**2. Intrinsic dimension of a flight path**

00:01 - 00:27

Consider this dataset with 2 features: latitude and longitude. These two features might track the flight of an airplane, for example. This dataset is 2-dimensional, yet it turns out that it can be closely approximated using only one feature: the displacement along the flight path. This dataset is intrinsically one-dimensional.

**3. Intrinsic dimension**

00:27 - 00:51

The intrinsic dimension of a dataset is the number of features required to approximate it. The intrinsic dimension informs dimension reduction, because it tells us how much a dataset can be compressed. In this video, you'll gain a solid understanding of the intrinsic dimension, and be able to use PCA to identify it in real-world datasets that have thousands of features.

**4. Versicolor dataset**

00:51 - 01:15

To better illustrate the intrinsic dimension, let's consider an example dataset containing only some of the samples from the iris dataset. Specifically, let's take three measurements from the iris versicolor samples: sepal length, sepal width, and petal width. So each sample is represented as a point in 3-dimensional space.

**5. Versicolor dataset has intrinsic dimension 2**

01:15 - 01:35

However, if we make a 3d scatter plot of the samples, we see that they all lie very close to a flat, 2-dimensional sheet. This means that the data can be approximated by using only two coordinates, without losing much information. So this dataset has intrinsic dimension 2.

**6. PCA identifies intrinsic dimension**

01:35 - 02:04

But scatter plots are only possible if there are 3 features or less. So how can the intrinsic dimension be identified, even if there are many features? This is where PCA is really helpful. The intrinsic dimension can be identified by counting the PCA features that have high variance. To see how, let's see what happens when PCA is applied to the dataset of versicolor samples.

**7. PCA of the versicolor samples**

02:04 - 02:16

PCA rotates and shifts the samples to align them with the coordinate axes. This expresses the samples using three PCA features.

**8. PCA features are ordered by variance descending**

02:16 - 02:47

The PCA features are in a special order. Here is a bar graph showing the variance of each of the PCA features. As you can see, each PCA feature has less variance than the last, and in this case the last PCA feature has very low variance. This agrees with the scatter plot of the PCA features, where the samples don't vary much in the vertical direction. In the other two directions, however, the variance is apparent.

**9. Variance and intrinsic dimension**

02:47 - 03:07

The intrinsic dimension is the number of PCA features that have significant variance. In our example, only the first two PCA features have significant variance. So this dataset has intrinsic dimension 2, which agrees with what we observed when inspecting the scatter plot.

**10. Plotting the variances of PCA features**

03:07 - 03:24

Let's see how to plot the variances of the PCA features in practice. Firstly, make the necessary imports. Then create a PCA model, and fit it to the samples. Now create a range enumerating the PCA features,

**11. Plotting the variances of PCA features**

03:24 - 03:31

and make a bar plot of the variances; the variances are available as the explained\_variance attribute of the PCA model.

**12. Intrinsic dimension can be ambiguous**

03:31 - 03:54

The intrinsic dimension is a useful idea that helps to guide dimension reduction. However, it is not always unambiguous. Here is a graph of the variances of the PCA features for the wine dataset. We could argue for an intrinsic dimension of 2, of 3, or even more, depending upon the threshold you chose.

**Dimension reduction with PCA**

Dimension reduction represents the same data using less features and is vital for building machine learning pipelines using real-world data. Finally, in this video, you'll learn how to perform dimension reduction using PCA.

**3. Dimension reduction with PCA**

00:16 - 00:33

We've seen already that the PCA features are in decreasing order of variance. PCA performs dimension reduction by discarding the PCA features with lower variance, which it assumes to be noise, and retaining the higher variance PCA features, which it assumes to be informative.

**4. Dimension reduction with PCA**

00:33 - 00:54

To use PCA for dimension reduction, you need to specify how many PCA features to keep. For example, specifying n\_components=2 when creating a PCA model tells it to keep only the first two PCA features. A good choice is the intrinsic dimension of the dataset, if you know it. Let's consider an example right away.

**5. Dimension reduction of iris dataset**

00:54 - 01:27

The iris dataset has 4 features representing the 4 measurements. Here, the measurements are in a numpy array called samples. Let's use PCA to reduce the dimension of the iris dataset to only 2. Begin by importing PCA as usual. Create a PCA model specifying n\_components=2, and then fit the model and transform the samples as usual. Printing the shape of the transformed samples, we see that there are only two features, as expected.

**6. Iris dataset in 2 dimensions**

01:27 - 01:55

Here is a scatterplot of the two PCA features, where the colors represent the three species of iris. Remarkably, despite having reduced the dimension from 4 to 2, the species can still be distinguished. Remember that PCA didn't even know that there were distinct species. PCA simply took the 2 PCA features with highest variance. As we can see, these two features are very informative.

**7. Dimension reduction with PCA**

01:55 - 02:12

PCA discards the low variance features, and assumes that the higher variance features are informative. Like all assumptions, there are cases where this doesn't hold. As we saw with the iris dataset, however, it often does in practice.

**8. Word frequency arrays**

02:12 - 02:44

In some cases, an alternative implementation of PCA needs to be used. Word frequency arrays are a great example. In a word-frequency array, each row corresponds to a document, and each column corresponds to a word from a fixed vocabulary. The entries of the word-frequency array measure how often each word appears in each document. Only some of the words from the vocabulary appear in any one document, so most entries of the word frequency array are zero.

**9. Sparse arrays and csr\_matrix**

02:44 - 02:58

Arrays like this are said to be "sparse", and are often represented using a special type of array called a "csr\_matrix". csr\_matrices save space by remembering only the non-zero entries of the array.

**10. TruncatedSVD and csr\_matrix**

02:58 - 03:20

Scikit-learn's PCA doesn't support csr\_matrices, and you'll need to use TruncatedSVD instead. TruncatedSVD performs the same transformation as PCA, but accepts csr matrices as input. Other than that, you interact with TruncatedSVD and PCA in exactly the same way.