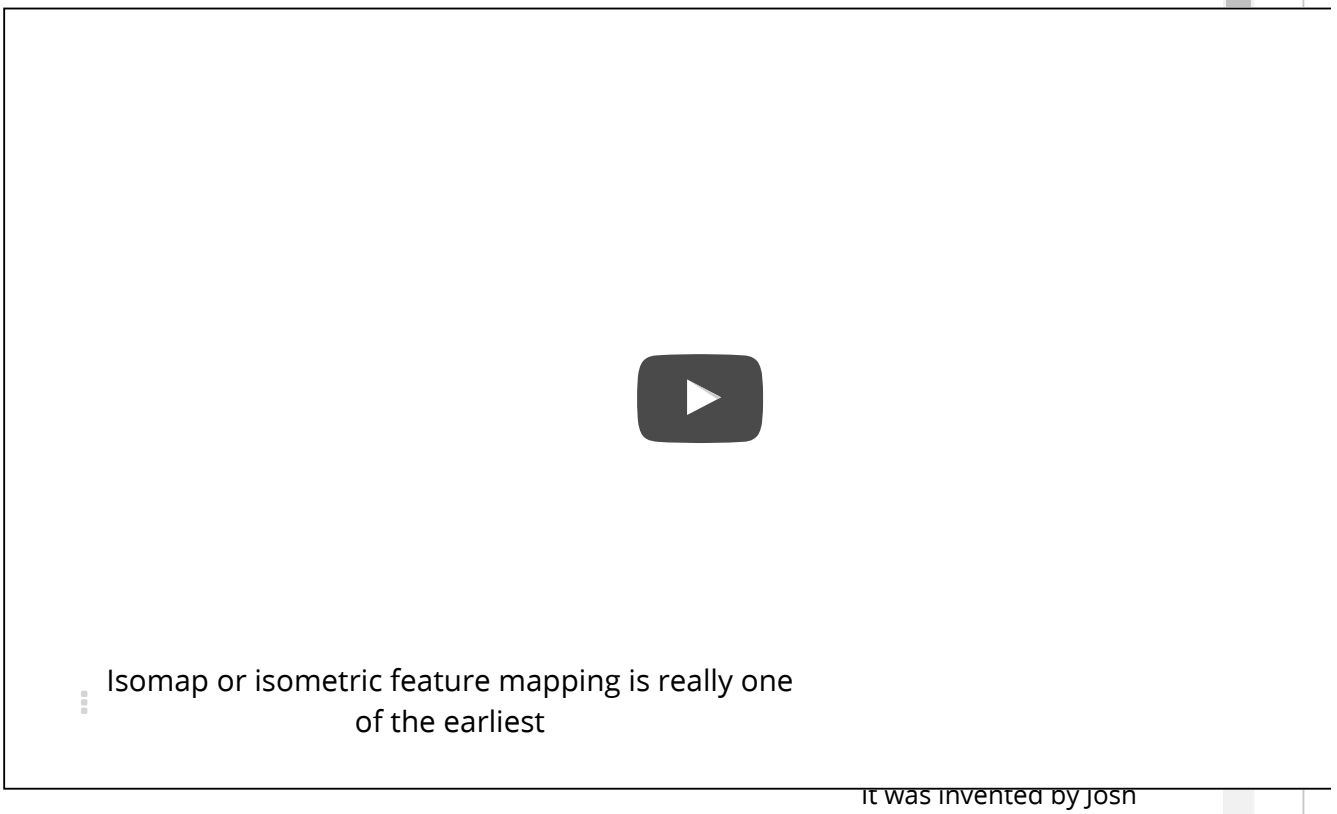
Video

Bookmark this page

SciKit-Learn and Isomap

Start of transcript. Skip to the end.



it was invented by Josh Tenenbaum and

two other professors actually around Y2K.

0:03 / 7:38 1.25x HD

Video

Download video file

Transcripts

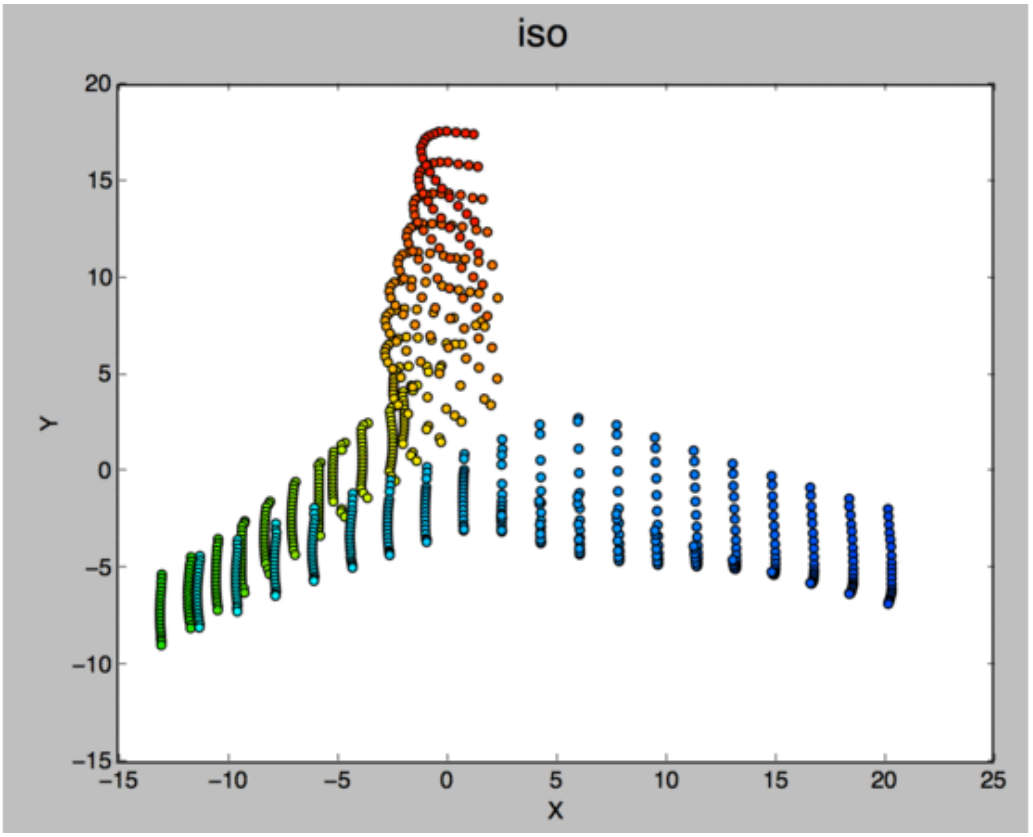
Download SubRip (.srt) file

Download Text (.txt) file

Isomap is a method of SciKit-Learn's manifold module:

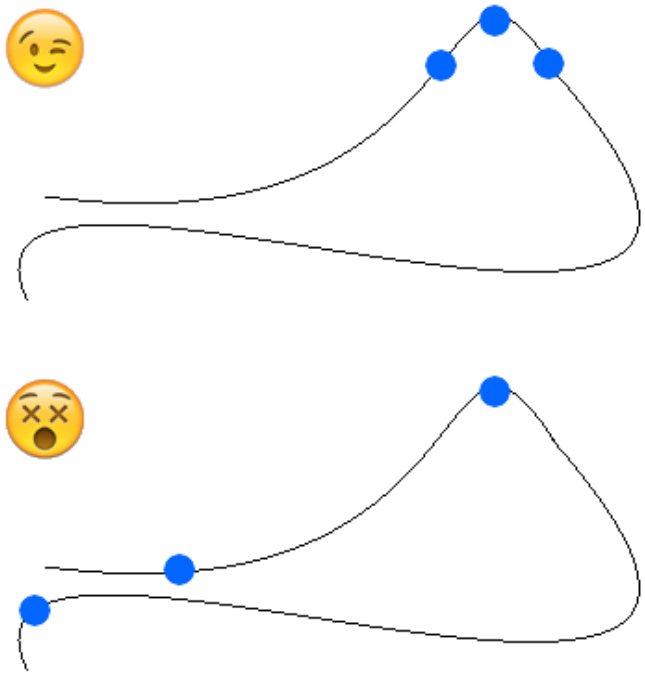
```
>>> from sklearn import manifold
>>> iso = manifold.Isomap(n_neighbors=4, n_components=2)
>>> iso.fit(df)
Isomap(eigen_solver='auto', max_iter=None, n_components=2, n_neighbors=4,
    neighbors_algorithm='auto', path_method='auto', tol=0)
```

As with PCA, `n_components` is the number of features you want your dataset projected onto, and `n_neighbors` defines the neighborhood size used to create the node neighborhood map. Be sure to experiment with n_neighbors particularly, as your neighborhood size will directly effect how the manifold mapping is generated:



The larger your `n_neighbors` value is, the longer it will take to calculate the node neighborhood map. In the above animation, the neighborhood sizes in order are: 2, 3, 4, 5, 6, 7, 8, 16, 32, 64. Notice that at 64 connectivity, the manifold almost looks similar to PCA. You will have to experiment with the neighborhood connectivity in order to get the best results.

That said, in addition to considering how many neighbors each sample should have, you also need to reflect on how many samples realistically even need to be collected in order for you to properly capture your lower dimensional manifold! A rule of thumb is the curvier your dataset is, and by curvier we mean the sharpness of the edges, the more dense your samples must be in order to capture its latent relationships:



With your Isomap instance created, you can calculate your new feature set the same way you did with PCA using `.fit()`. Fitting the data just calculates the basis of the lower dimensional encoding. To actually project your data into that space, transforming it by calling:

```
>>> manifold = iso.transform(df)
>>> df.shape
(430, 6)

>>> manifold.shape
(430, 2)
```

Unlike PCA, Isomap transformations are unidirectional so you will not be able to `.inverse_transform()` your projected data back into your original feature space, even if it has the same number of dimensions as your original dataset.

English