nonlinear

April 22, 2021

1 Nonlinear Dimensionality Reduction

This notebook will take a look at a modified subset of the COIL-20 dataset. This is a collection of 20 objects, each of which was photographed on a turntable at 5^o rotational incements. Our updated version of the dataset introduces versions of the images for each viewing angle. As a result, the collection of images for each object lies on a two-dimensional manifold: one dimension corresponding to viewing angle and one corresponding to image rotation.

[4]:
```
# Handle imports and load the data set.

import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from matplotlib import offsetbox
from mpl_toolkits.axes_grid1 import ImageGrid
from sklearn.manifold import Isomap
from sklearn.manifold import TSNE


#plt.rcParams['figure.figsize'] = [12, 8]
#plt.rcParams['figure.dpi'] = 100


ducks = np.load('ducks.npy')
cats = np.load('cats.npy')
multi = np.load('all.npy')
```

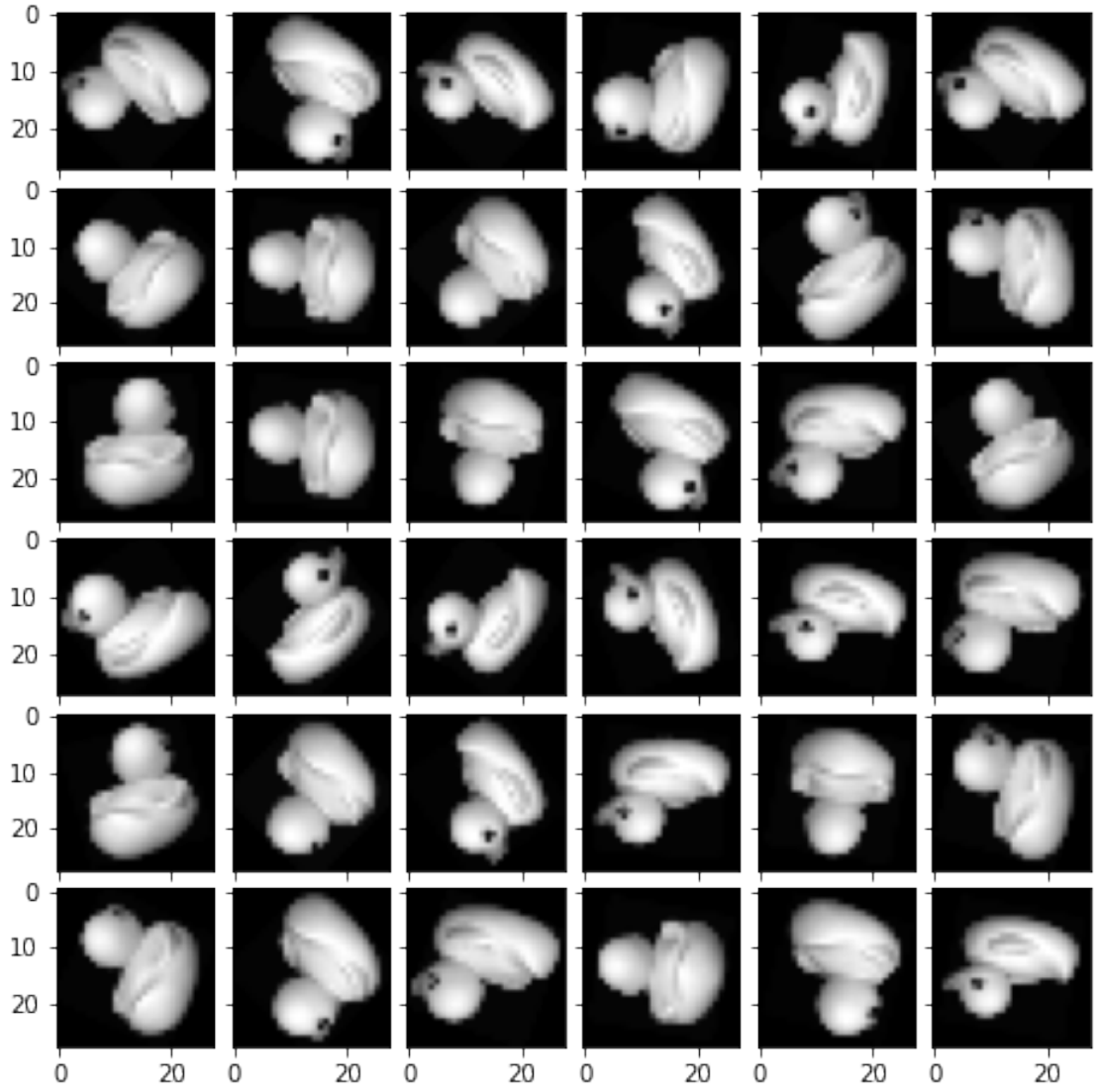
Here is a random subset of the duck images:

[2]:
```
choices = np.random.choice(np.arange(ducks.shape[0]), 36)
duck_images = ducks.reshape(-1, 28, 28)[choices, :,:]
print(duck_images.shape)

fig = plt.figure(figsize=(8., 8.))
grid = ImageGrid(fig, 111,
                 nrows_ncols=(6, 6),
                 axes_pad=0.1,  # pad between axes in inch.
                 )
```

```
for ax, im in zip(grid, duck_images):
    ax.imshow(im, cmap='gray')
```

```
(36, 28, 28)
```



[5]:
```
# https://jakevdp.github.io/PythonDataScienceHandbook/05.10-manifold-learning.
↪html
def plot_components(data, model, images=None, ax=None,
                    thumb_frac=0.05, cmap='gray'):
    """ Plot a 2d-projection of the image data along with a subset of the␣
↪images."""
```

```
    ax = ax or plt.gca()

    proj = model.fit_transform(data)
    ax.plot(proj[:, 0], proj[:, 1], '.k')

    if images is not None:
        min_dist_2 = (thumb_frac * max(proj.max(0) - proj.min(0))) ** 2
        shown_images = np.array([2 * proj.max(0)])
        for i in range(data.shape[0]):
            dist = np.sum((proj[i] - shown_images) ** 2, 1)
            if np.min(dist)  < min_dist_2:
                # don't show points that are too close
                continue
            shown_images = np.vstack([shown_images, proj[i]])
            imagebox = offsetbox.AnnotationBbox(
                offsetbox.OffsetImage(images[i], cmap=cmap),
                                      proj[i])
            ax.add_artist(imagebox)
```

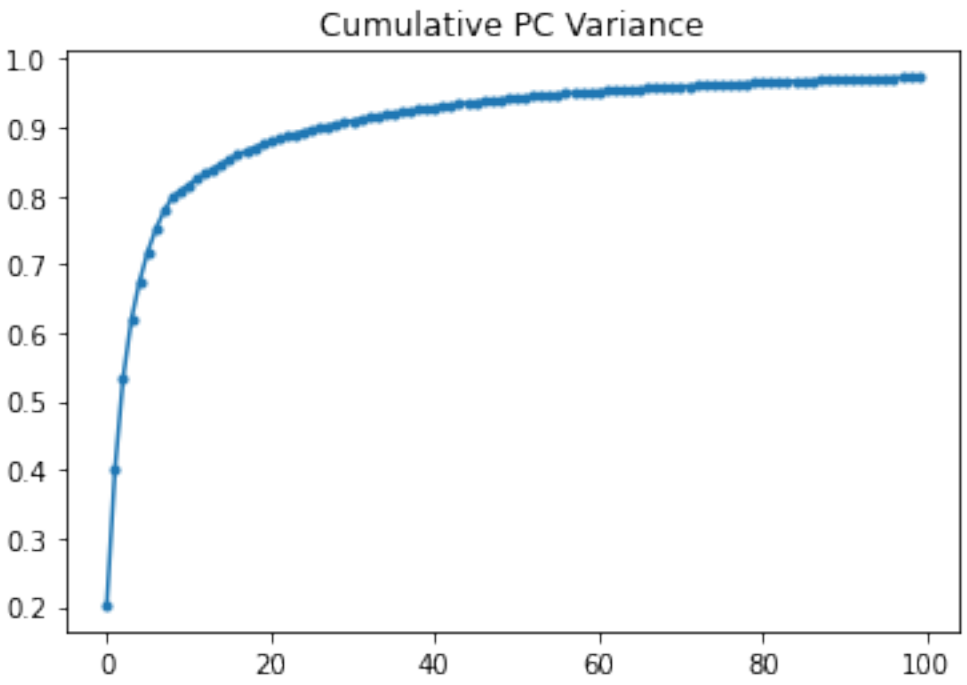
1.1 PCA

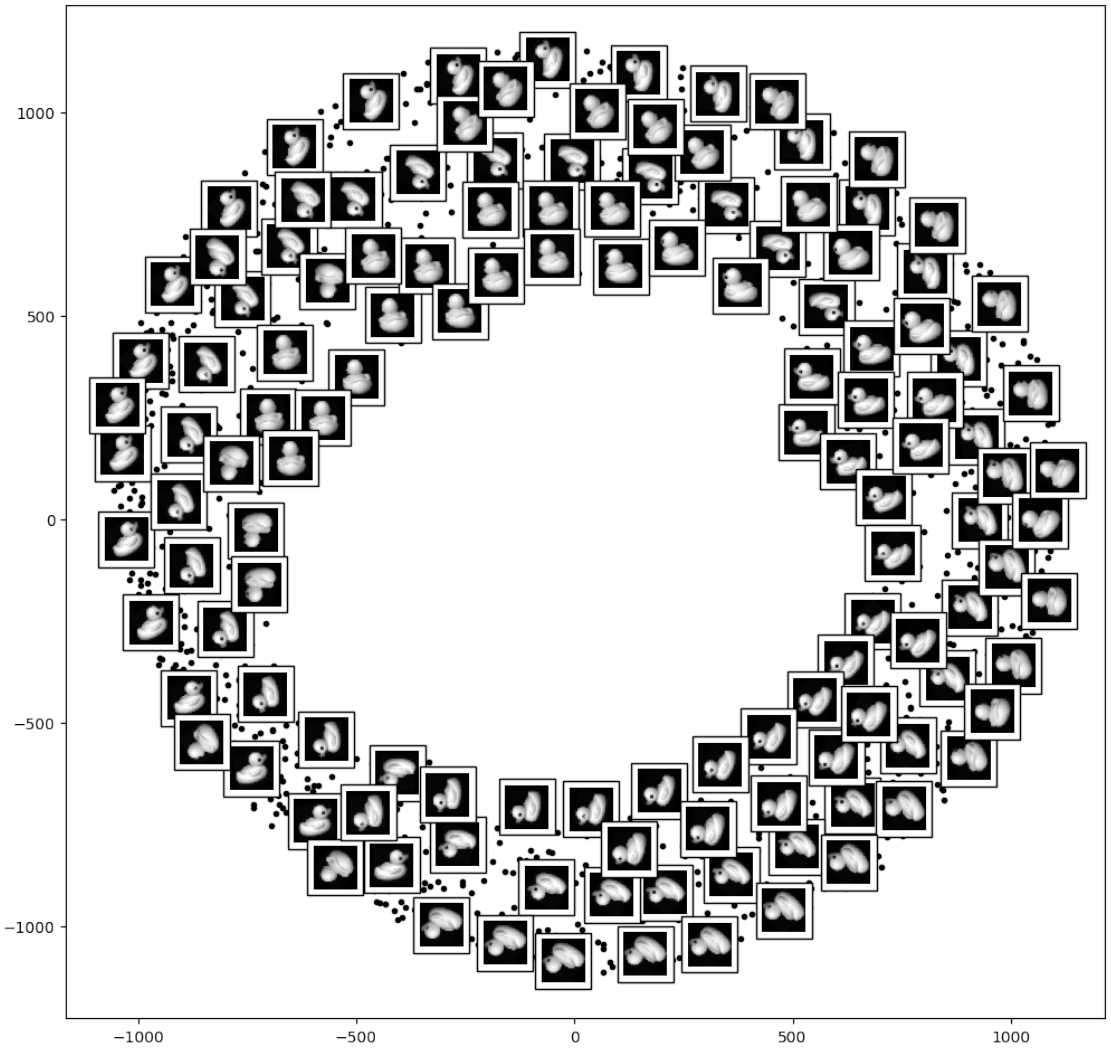
First let's see how PCA handles this data...

[6]:
```
pca = PCA()
pca.fit(ducks)

plt.plot(np.cumsum(pca.explained_variance_ratio_[0:100]), '.-')
plt.title('Cumulative PC Variance')

plt.figure(figsize=(12,12), dpi=100)
plot_components(ducks,
                model=PCA(n_components=2),
                images=ducks.reshape((-1, 28, 28)))
```

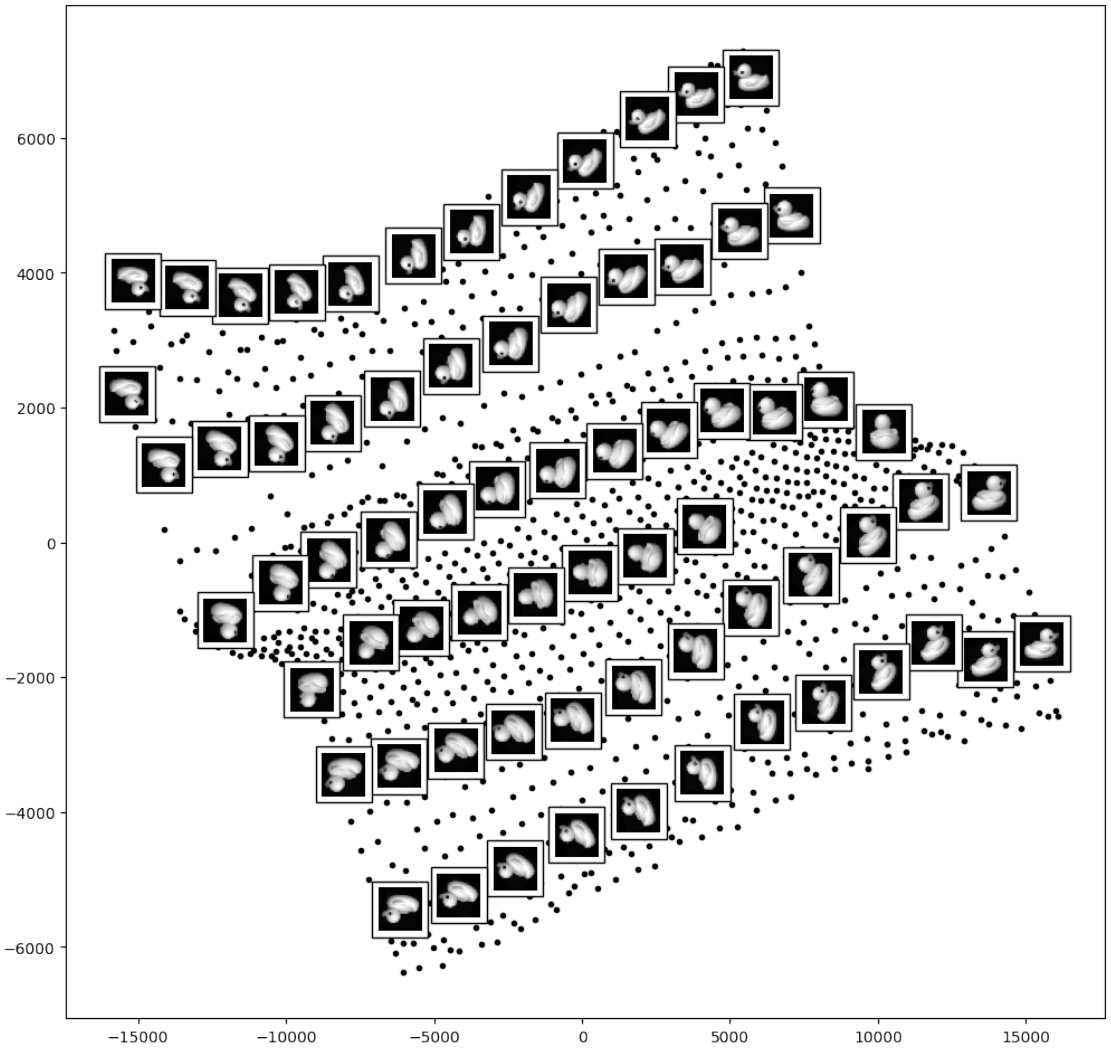

Cumulative PC Variance
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100



1.2 ISOMAP

The first two principal components only capture 40% of the variance, and the projected points don't seem to capture the interesting directions of variation on the manifold. Let's see if ISOMAP does better…

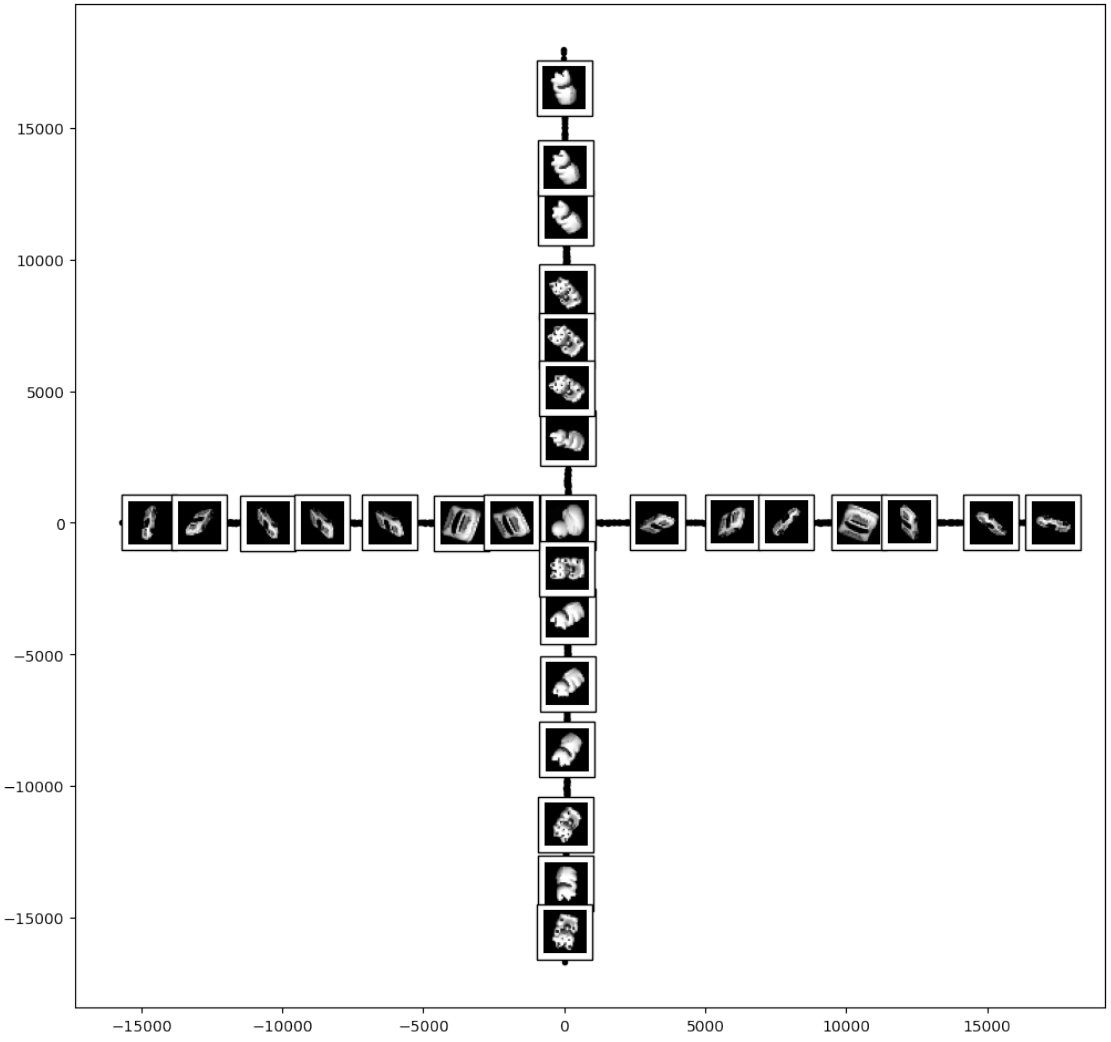
[7]:
```
plt.figure(figsize=(12,12), dpi=100)
plot_components(ducks,
                model=Isomap(n_components=2),
                images=ducks.reshape((-1, 28, 28)))
```



Much better! ISOMAP cleanly captures the to dimensions that we expected to see.

Let's see what happens if we repeat the experminent on a dataset that includes three different objects. In this case we expect that the data will lie on three separate 2D manifolds embedded in the 784-dimensional space…

[8]:
```
plt.figure(figsize=(12,12), dpi=100)
plot_components(multi,
                model=Isomap(n_components=2),
                images=multi.reshape((-1, 28, 28)))
```

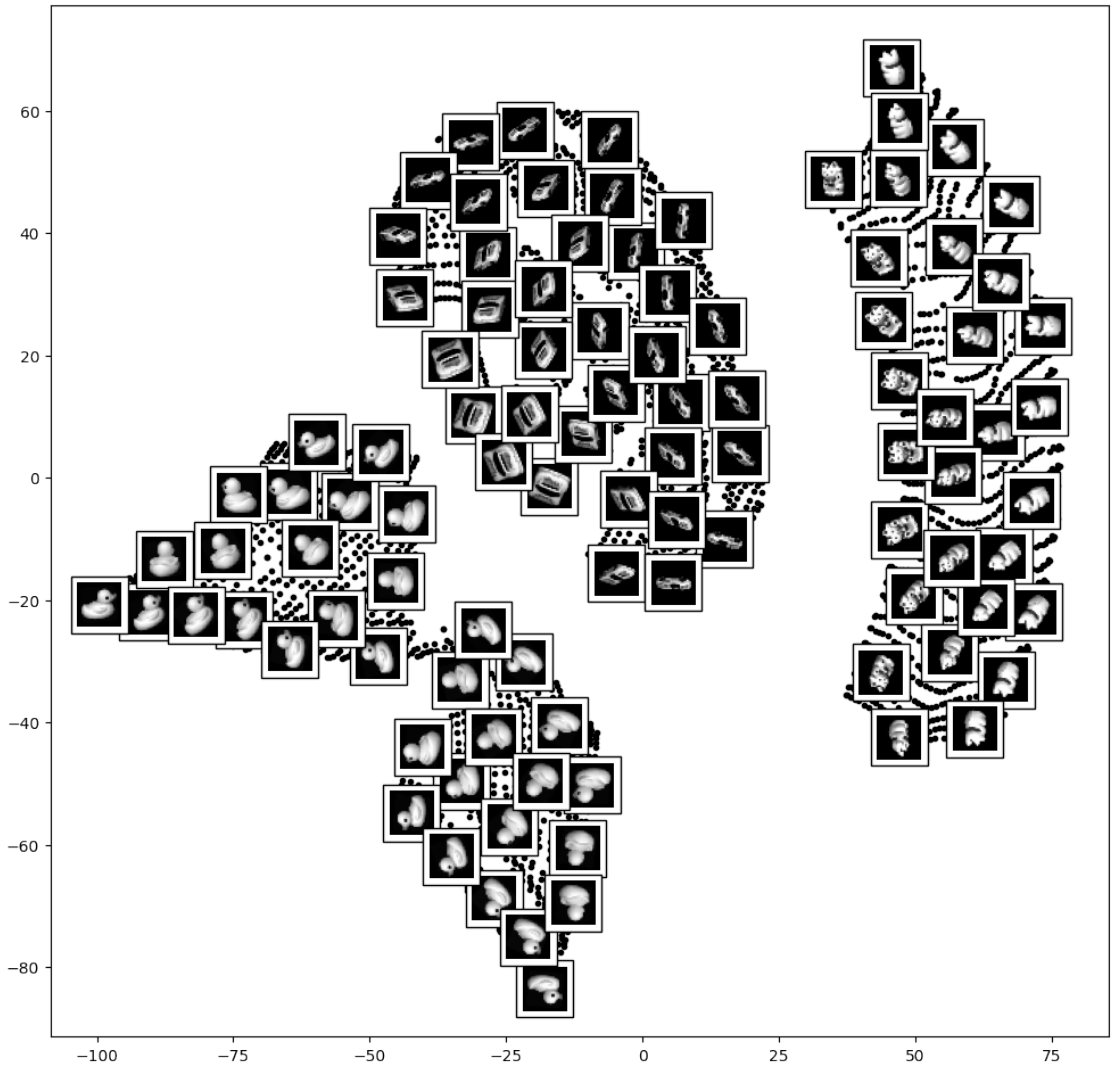


[ ]:
```
## t-SNE

The results were not great!
```

[9]:
```
plt.figure(figsize=(12,12), dpi=100)

plot_components(multi,
                model=TSNE(n_components=2),
                images=multi.reshape((-1, 28, 28)))
```


60
40
20
0
−20
−40
−60
−80
−100
−75
−50
−25
0
25
50
75

[ ]: