We could then go on to classify this data, perhaps using manifold features as inputs to the classification algorithm as we did in "In-Depth: Support Vector Machines" on page 405.

Example: Visualizing Structure in Digits

As another example of using manifold learning for visualization, let's take a look at the MNIST handwritten digits set. This data is similar to the digits we saw in "In-Depth: Decision Trees and Random Forests" on page 421, but with many more pixels per image. It can be downloaded from *http://mldata.org/* with the Scikit-Learn utility:

```
In[22]: from sklearn.datasets import fetch_mldata
        mnist = fetch_mldata('MNIST original')
        mnist.data.shape

Out[22]: (70000, 784)
```

This consists of 70,000 images, each with 784 pixels (i.e., the images are 28×28). As before, we can take a look at the first few images (Figure 5-107):

```
In[23]: fig, ax = plt.subplots(6, 8, subplot_kw=dict(xticks=[], yticks=[]))
        for i, axi in enumerate(ax.flat):
            axi.imshow(mnist.data[1250 * i].reshape(28, 28), cmap='gray_r')
```

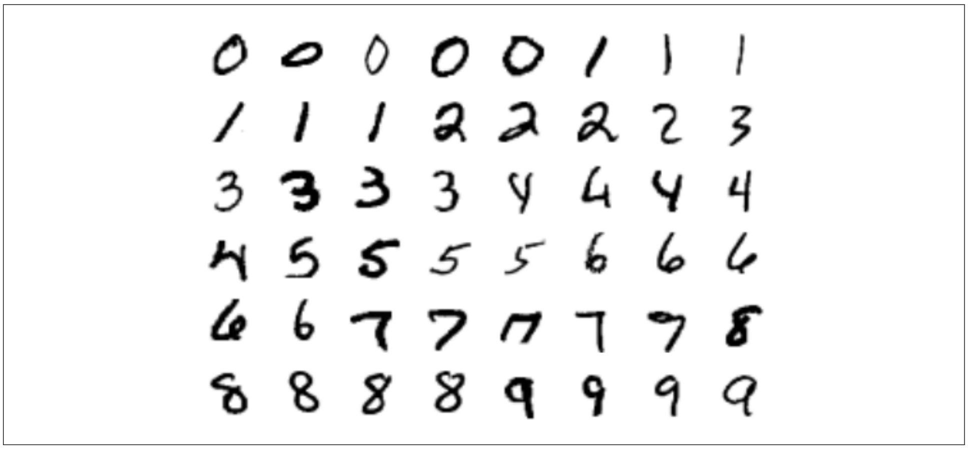
Figure 5-107. Examples of the MNIST digits

This gives us an idea of the variety of handwriting styles in the dataset.

Let's compute a manifold learning projection across the data, illustrated in Figure 5-108. For speed here, we'll only use 1/30 of the data, which is about ~2,000 points (because of the relatively poor scaling of manifold learning, I find that a few thousand samples is a good number to start with for relatively quick exploration before moving to a full calculation):

```
In[24]:
# use only 1/30 of the data: full dataset takes a long time!
data = mnist.data[::30]
target = mnist.target[::30]

model = Isomap(n_components=2)
proj = model.fit_transform(data)
plt.scatter(proj[:, 0], proj[:, 1], c=target, cmap=plt.cm.get_cmap('jet', 10))
plt.colorbar(ticks=range(10))
plt.clim(-0.5, 9.5);
```

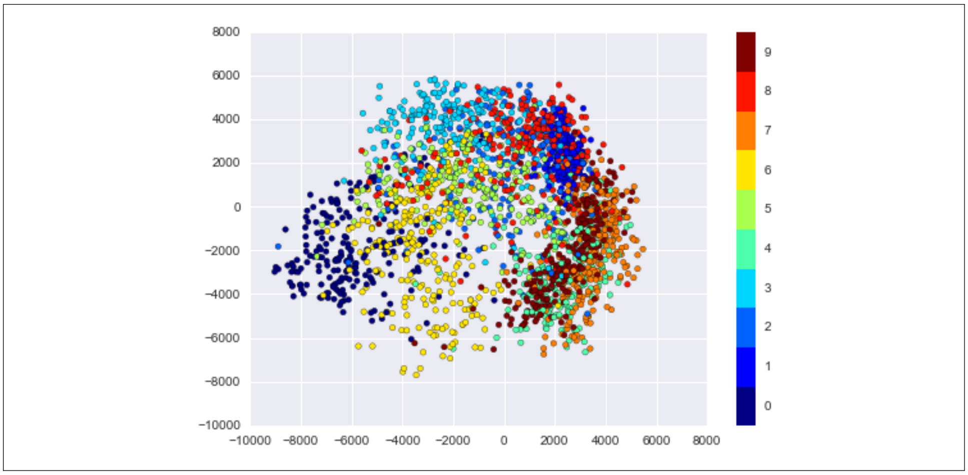

Figure 5-108. Isomap embedding of the MNIST digit data

The resulting scatter plot shows some of the relationships between the data points, but is a bit crowded. We can gain more insight by looking at just a single number at a time (Figure 5-109):

```
In[25]: from sklearn.manifold import Isomap

        # Choose 1/4 of the "1" digits to project
        data = mnist.data[mnist.target == 1][::4]

        fig, ax = plt.subplots(figsize=(10, 10))
        model = Isomap(n_neighbors=5, n_components=2, eigen_solver='dense')
        plot_components(data, model, images=data.reshape((-1, 28, 28)),
                        ax=ax, thumb_frac=0.05, cmap='gray_r')
```

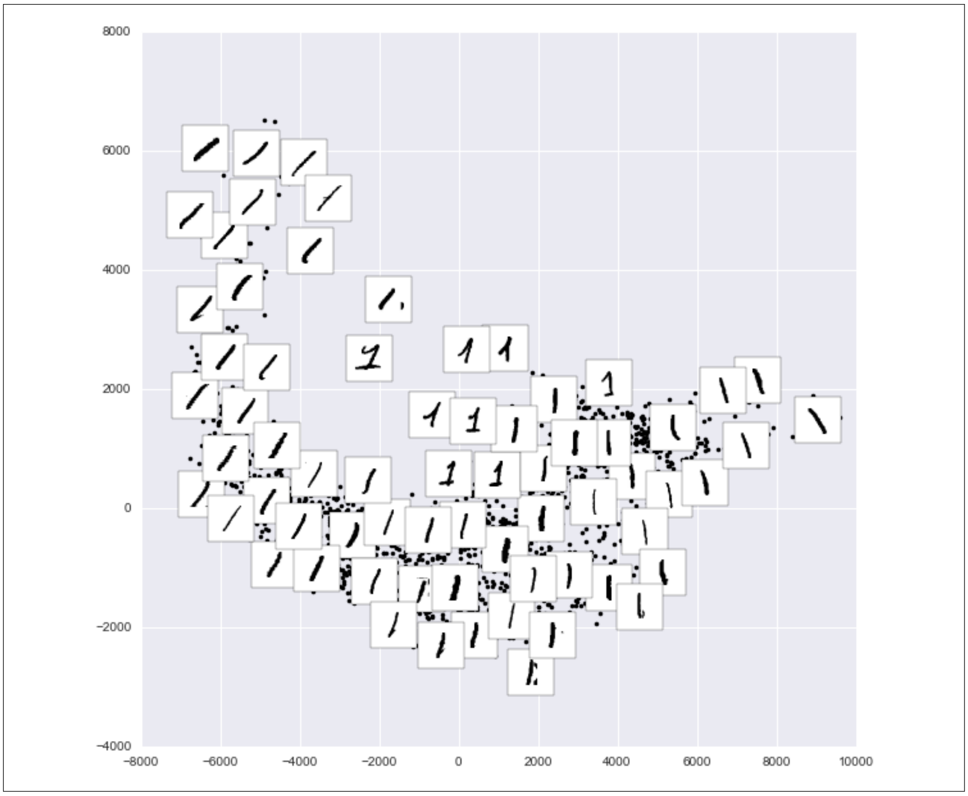

Figure 5-109. Isomap embedding of only the 1s within the digits data

The result gives you an idea of the variety of forms that the number “1” can take within the dataset. The data lies along a broad curve in the projected space, which appears to trace the orientation of the digit. As you move up the plot, you find ones that have hats and/or bases, though these are very sparse within the dataset. The projection lets us identify outliers that have data issues (i.e., pieces of the neighboring digits that snuck into the extracted images).

Now, this in itself may not be useful for the task of classifying digits, but it does help us get an understanding of the data, and may give us ideas about how to move forward, such as how we might want to preprocess the data before building a classification pipeline.

In Depth: k-Means Clustering

In the previous few sections, we have explored one category of unsupervised machine learning models: dimensionality reduction. Here we will move on to another class of unsupervised machine learning models: clustering algorithms. Clustering algorithms