Example: Handwritten Digits

For an example of where this might be useful, let's look at an interesting visualization of some handwritten digits data. This data is included in Scikit-Learn, and consists of nearly 2,000 8×8 thumbnails showing various handwritten digits.

For now, let's start by downloading the digits data and visualizing several of the example images with `plt.imshow()` (Figure 4-57):

```
In[12]: # load images of the digits 0 through 5 and visualize several of them
        from sklearn.datasets import load_digits
        digits = load_digits(n_class=6)

        fig, ax = plt.subplots(8, 8, figsize=(6, 6))
        for i, axi in enumerate(ax.flat):
            axi.imshow(digits.images[i], cmap='binary')
            axi.set(xticks=[], yticks=[])
```

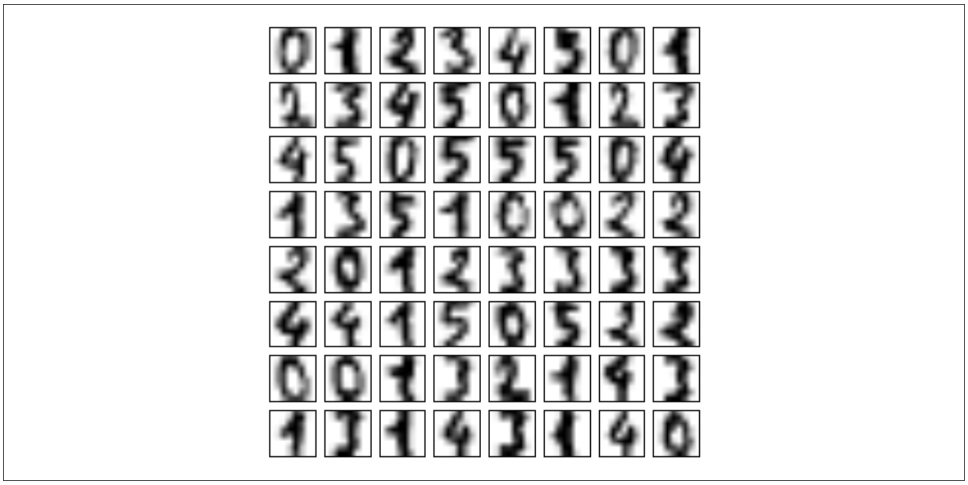

Figure 4-57. Sample of handwritten digit data

Because each digit is defined by the hue of its 64 pixels, we can consider each digit to be a point lying in 64-dimensional space: each dimension represents the brightness of one pixel. But visualizing relationships in such high-dimensional spaces can be extremely difficult. One way to approach this is to use a *dimensionality reduction* technique such as manifold learning to reduce the dimensionality of the data while maintaining the relationships of interest. Dimensionality reduction is an example of unsupervised machine learning, and we will discuss it in more detail in "What Is Machine Learning?" on page 332.

Deferring the discussion of these details, let's take a look at a two-dimensional manifold learning projection of this digits data (see "In-Depth: Manifold Learning" on page 445 for details):

```
In[13]: # project the digits into 2 dimensions using IsoMap
        from sklearn.manifold import Isomap
        iso = Isomap(n_components=2)
        projection = iso.fit_transform(digits.data)
```

We'll use our discrete colormap to view the results, setting the `ticks` and `clim` to improve the aesthetics of the resulting colorbar (Figure 4-58):

```
In[14]: # plot the results
        plt.scatter(projection[:, 0], projection[:, 1], lw=0.1,
                    c=digits.target, cmap=plt.cm.get_cmap('cubehelix', 6))
        plt.colorbar(ticks=range(6), label='digit value')
        plt.clim(-0.5, 5.5)
```

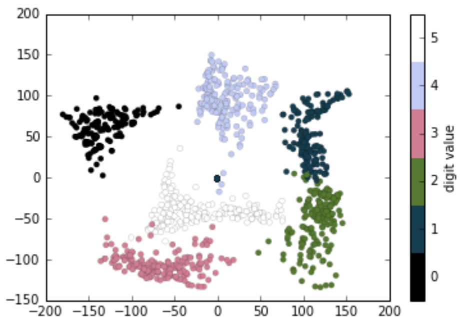

Figure 4-58. Manifold embedding of handwritten digit pixels

The projection also gives us some interesting insights on the relationships within the dataset: for example, the ranges of 5 and 3 nearly overlap in this projection, indicating that some handwritten fives and threes are difficult to distinguish, and therefore more likely to be confused by an automated classification algorithm. Other values, like 0 and 1, are more distantly separated, and therefore much less likely to be confused. This observation agrees with our intuition, because 5 and 3 look much more similar than do 0 and 1.

We'll return to manifold learning and digit classification in Chapter 5.

Multiple Subplots

Sometimes it is helpful to compare different views of data side by side. To this end, Matplotlib has the concept of *subplots*: groups of smaller axes that can exist together within a single figure. These subplots might be insets, grids of plots, or other more complicated layouts. In this section, we'll explore four routines for creating subplots in Matplotlib. We'll start by setting up the notebook for plotting and importing the functions we will use: