Example: Selecting Random Points

One common use of fancy indexing is the selection of subsets of rows from a matrix. For example, we might have an *N* by *D* matrix representing *N* points in *D* dimensions, such as the following points drawn from a two-dimensional normal distribution:

```
In[13]: mean = [0, 0]
        cov = [[1, 2],
               [2, 5]]
        X = rand.multivariate_normal(mean, cov, 100)
        X.shape

Out[13]: (100, 2)
```

Using the plotting tools we will discuss in Chapter 4, we can visualize these points as a scatter plot (Figure 2-7):

```
In[14]: %matplotlib inline
        import matplotlib.pyplot as plt
        import seaborn; seaborn.set()  # for plot styling

        plt.scatter(X[:, 0], X[:, 1]);
```

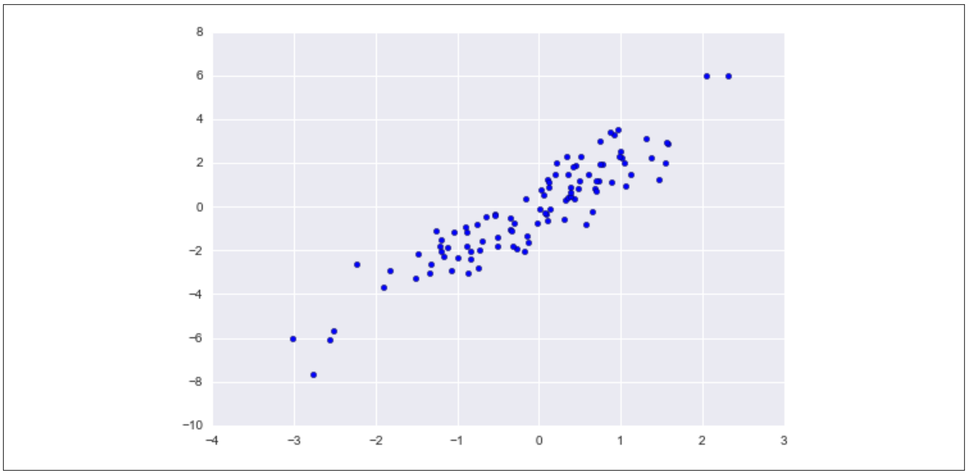

Figure 2-7. Normally distributed points

Let's use fancy indexing to select 20 random points. We'll do this by first choosing 20 random indices with no repeats, and use these indices to select a portion of the original array:

```
In[15]: indices = np.random.choice(X.shape[0], 20, replace=False)
        indices

Out[15]: array([93, 45, 73, 81, 50, 10, 98, 94,  4, 64, 65, 89, 47, 84, 82,
                80, 25, 90, 63, 20])
```

```
In[16]: selection = X[indices]  # fancy indexing here
        selection.shape

Out[16]: (20, 2)
```

Now to see which points were selected, let's over-plot large circles at the locations of the selected points (Figure 2-8):

```
In[17]: plt.scatter(X[:, 0], X[:, 1], alpha=0.3)
        plt.scatter(selection[:, 0], selection[:, 1],
                    facecolor='none', s=200);
```

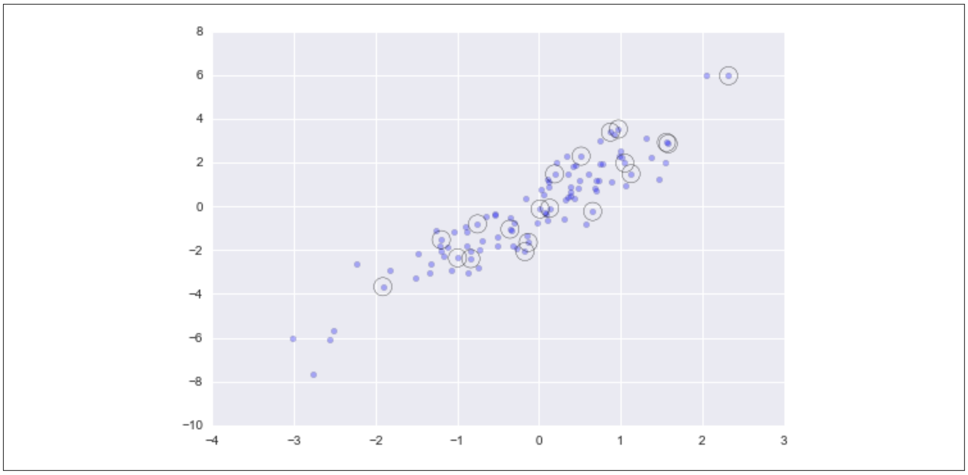

Figure 2-8. Random selection among points

This sort of strategy is often used to quickly partition datasets, as is often needed in train/test splitting for validation of statistical models (see "Hyperparameters and Model Validation" on page 359), and in sampling approaches to answering statistical questions.

Modifying Values with Fancy Indexing

Just as fancy indexing can be used to access parts of an array, it can also be used to modify parts of an array. For example, imagine we have an array of indices and we'd like to set the corresponding items in an array to some value:

```
In[18]: x = np.arange(10)
        i = np.array([2, 1, 8, 4])
        x[i] = 99
        print(x)

[ 0 99 99  3 99  5  6  7 99  9]
```

We can use any assignment-type operator for this. For example: