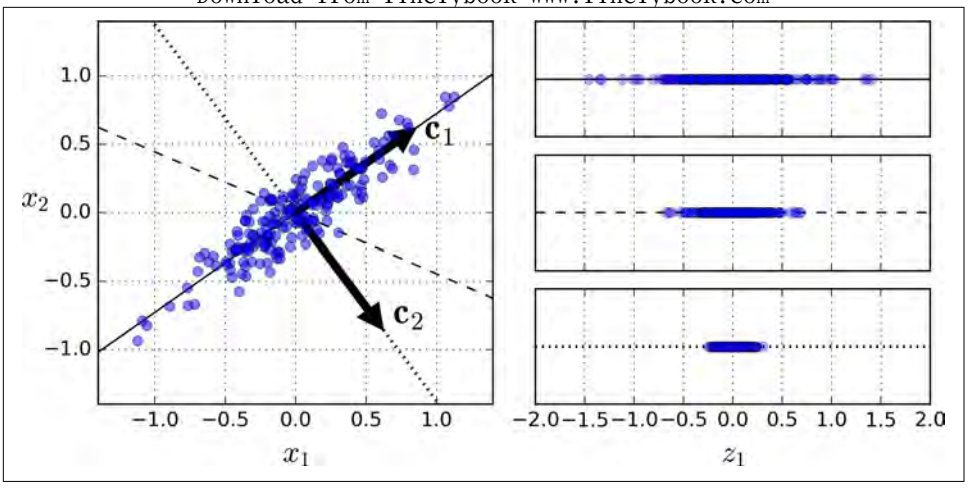

Figure 8-7. Selecting the subspace onto which to project

It seems reasonable to select the axis that preserves the maximum amount of variance, as it will most likely lose less information than the other projections. Another way to justify this choice is that it is the axis that minimizes the mean squared distance between the original dataset and its projection onto that axis. This is the rather simple idea behind PCA.[4]

Principal Components

PCA identifies the axis that accounts for the largest amount of variance in the training set. In Figure 8-7, it is the solid line. It also finds a second axis, orthogonal to the first one, that accounts for the largest amount of remaining variance. In this 2D example there is no choice: it is the dotted line. If it were a higher-dimensional dataset, PCA would also find a third axis, orthogonal to both previous axes, and a fourth, a fifth, and so on—as many axes as the number of dimensions in the dataset.

The unit vector that defines the i^{th} axis is called the i^{th} *principal component* (PC). In Figure 8-7, the 1^{st} PC is $\mathbf{c}_1$ and the 2^{nd} PC is $\mathbf{c}_2$. In Figure 8-2 the first two PCs are represented by the orthogonal arrows in the plane, and the third PC would be orthogonal to the plane (pointing up or down).

4 "On Lines and Planes of Closest Fit to Systems of Points in Space," K. Pearson (1901).

The direction of the principal components is not stable: if you perturb the training set slightly and run PCA again, some of the new PCs may point in the opposite direction of the original PCs. However, they will generally still lie on the same axes. In some cases, a pair of PCs may even rotate or swap, but the plane they define will generally remain the same.

So how can you find the principal components of a training set? Luckily, there is a standard matrix factorization technique called *Singular Value Decomposition* (SVD) that can decompose the training set matrix $\mathbf{X}$ into the dot product of three matrices $\mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T$, where $\mathbf{V}^T$ contains all the principal components that we are looking for, as shown in Equation 8-1.

Equation 8-1. Principal components matrix

$$\mathbf{V}^T = \begin{pmatrix} | & | & & | \\ \mathbf{c_1} & \mathbf{c_2} & \cdots & \mathbf{c_n} \\ | & | & & | \end{pmatrix}$$

The following Python code uses NumPy's `svd()` function to obtain all the principal components of the training set, then extracts the first two PCs:

```
X_centered = X - X.mean(axis=0)
U, s, V = np.linalg.svd(X_centered)
c1 = V.T[:, 0]
c2 = V.T[:, 1]
```

PCA assumes that the dataset is centered around the origin. As we will see, Scikit-Learn's PCA classes take care of centering the data for you. However, if you implement PCA yourself (as in the preceding example), or if you use other libraries, don't forget to center the data first.

Projecting Down to d Dimensions

Once you have identified all the principal components, you can reduce the dimensionality of the dataset down to d dimensions by projecting it onto the hyperplane defined by the first d principal components. Selecting this hyperplane ensures that the projection will preserve as much variance as possible. For example, in Figure 8-2 the 3D dataset is projected down to the 2D plane defined by the first two principal components, preserving a large part of the dataset's variance. As a result, the 2D projection looks very much like the original 3D dataset.

To project the training set onto the hyperplane, you can simply compute the dot product of the training set matrix $\mathbf{X}$ by the matrix $\mathbf{W}_d$, defined as the matrix contain-