

The following code produces the model represented on the left of [Figure 5-11](#) using Scikit-Learn's SVR class (which supports the kernel trick). The SVR class is the regression equivalent of the SVC class, and the LinearSVR class is the regression equivalent of the LinearSVC class. The LinearSVR class scales linearly with the size of the training set (just like the LinearSVC class), while the SVR class gets much too slow when the training set grows large (just like the SVC class).

```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```



SVMs can also be used for outlier detection; see Scikit-Learn's documentation for more details.

Under the Hood

This section explains how SVMs make predictions and how their training algorithms work, starting with linear SVM classifiers. You can safely skip it and go straight to the exercises at the end of this chapter if you are just getting started with Machine Learning, and come back later when you want to get a deeper understanding of SVMs.

First, a word about notations: in [Chapter 4](#) we used the convention of putting all the model parameters in one vector θ , including the bias term θ_0 and the input feature weights θ_1 to θ_n , and adding a bias input $x_0 = 1$ to all instances. In this chapter, we will use a different convention, which is more convenient (and more common) when you are dealing with SVMs: the bias term will be called b and the feature weights vector will be called \mathbf{w} . No bias feature will be added to the input feature vectors.

Decision Function and Predictions

The linear SVM classifier model predicts the class of a new instance \mathbf{x} by simply computing the decision function $\mathbf{w}^T \cdot \mathbf{x} + b = w_1 x_1 + \dots + w_n x_n + b$: if the result is positive, the predicted class \hat{y} is the positive class (1), or else it is the negative class (0); see [Equation 5-2](#).

Equation 5-2. Linear SVM classifier prediction

$$\hat{y} = \begin{cases} 0 & \text{if } \mathbf{w}^T \cdot \mathbf{x} + b < 0, \\ 1 & \text{if } \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases}$$

Figure 5-12 shows the decision function that corresponds to the model on the right of Figure 5-4: it is a two-dimensional plane since this dataset has two features (petal width and petal length). The decision boundary is the set of points where the decision function is equal to 0: it is the intersection of two planes, which is a straight line (represented by the thick solid line).³

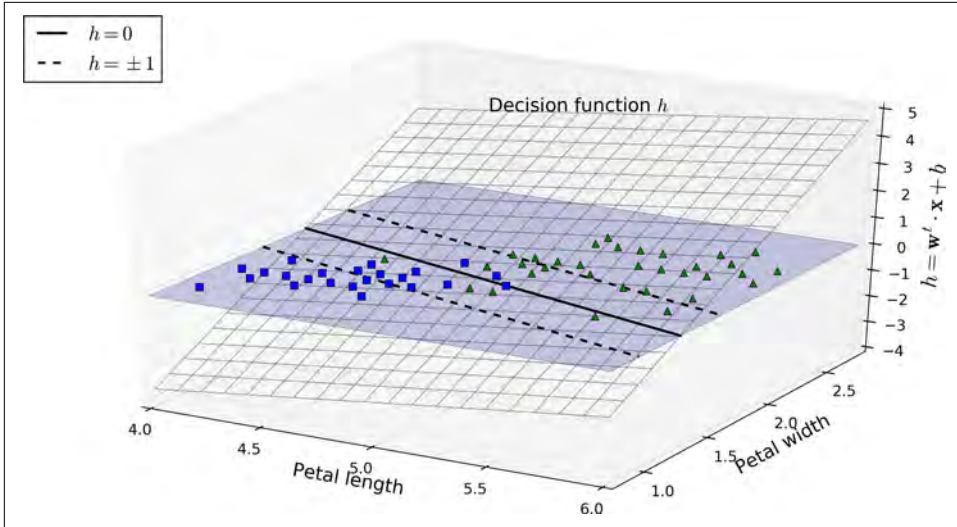


Figure 5-12. Decision function for the iris dataset

The dashed lines represent the points where the decision function is equal to 1 or -1: they are parallel and at equal distance to the decision boundary, forming a margin around it. Training a linear SVM classifier means finding the value of \mathbf{w} and b that make this margin as wide as possible while avoiding margin violations (hard margin) or limiting them (soft margin).

Training Objective

Consider the slope of the decision function: it is equal to the norm of the weight vector, $\|\mathbf{w}\|$. If we divide this slope by 2, the points where the decision function is equal to ± 1 are going to be twice as far away from the decision boundary. In other words, dividing the slope by 2 will multiply the margin by 2. Perhaps this is easier to visualize in 2D in Figure 5-13. The smaller the weight vector \mathbf{w} , the larger the margin.

³ More generally, when there are n features, the decision function is an n -dimensional *hyperplane*, and the decision boundary is an $(n - 1)$ -dimensional hyperplane.