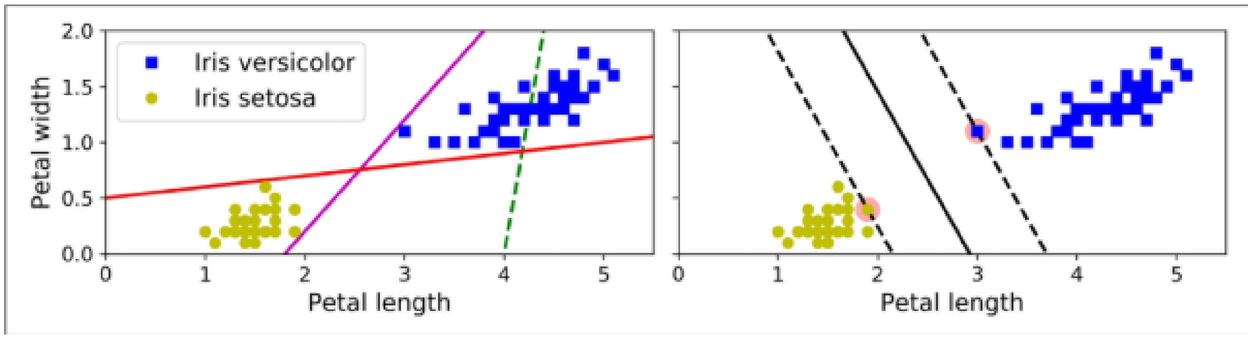
Chapter 5

Support Vector Machines

Support Vector Machines (SVMs) are powerful and versatile models used for both classification and regression. The core idea of SVMs is best understood visually. In the case of linearly separable data, many linear classifiers can separate two classes, but SVMs select the decision boundary that maximizes the margin, meaning the distance between the boundary and the closest training points from each class. This concept is known as large margin classification, and it leads to models that tend to generalize better to unseen data.



/ *Figure 5-1. Large margin classification*

In large margin classification, the decision boundary is determined only by the training instances that lie closest to it. These critical instances are called support vectors. Points that are far away from the margin do not influence the model, which explains why adding more training data “off the street” usually does not change the decision boundary. This property makes SVMs robust and efficient once the margin is defined.

Because margin computation relies heavily on distances, SVMs are highly sensitive to the scale of input features. If features are measured on very different scales, the margin geometry becomes distorted, leading to poor decision boundaries. For this reason, feature scaling—most commonly standardization—is an essential preprocessing step when training SVMs.

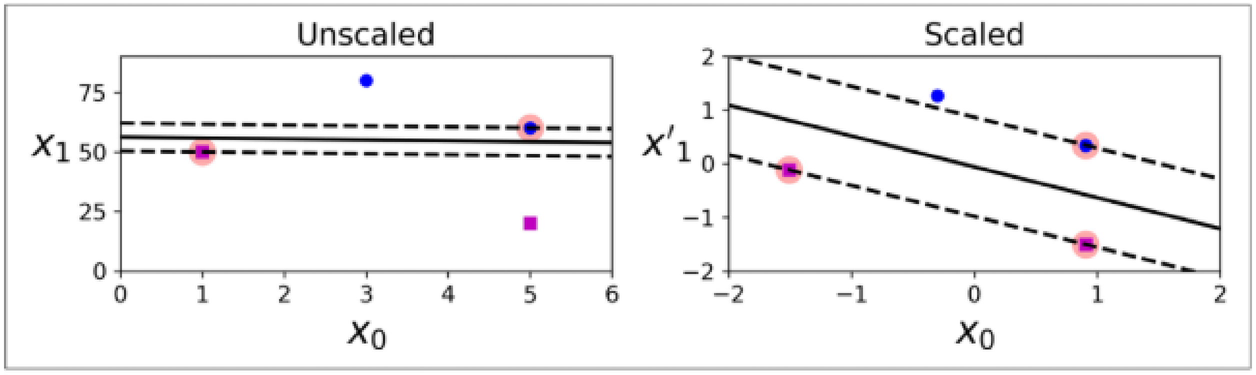


Figure 5-2. Sensitivity to feature scales

If SVMs are trained with the strict requirement that all training instances must lie outside the margin and be correctly classified, the model is referred to as a hard margin classifier. While theoretically elegant, hard margin classification has two major drawbacks: it only works when the data is perfectly linearly separable, and it is extremely sensitive to outliers. Even a single outlier can drastically alter or invalidate the decision boundary.

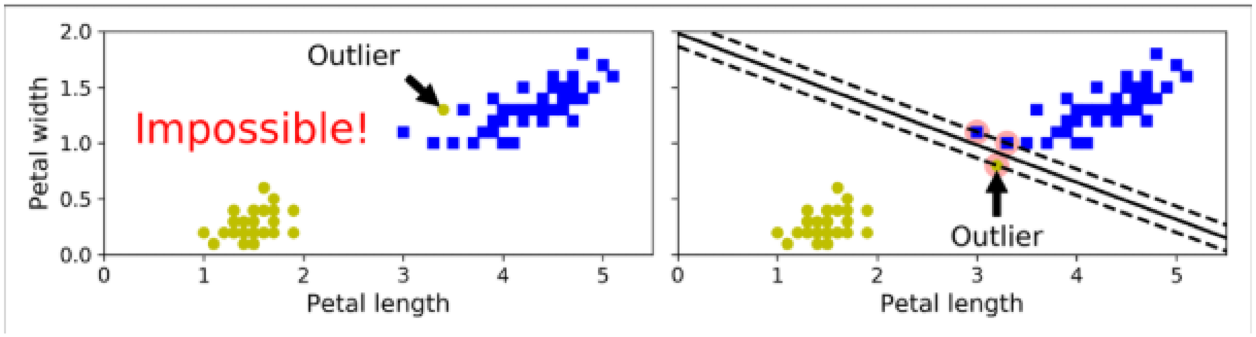


Figure 5-3. Hard margin sensitivity to outliers

To overcome these limitations, SVMs typically use **soft margin classification**, which allows some margin violations in exchange for better generalization. This trade-off is controlled by the hyperparameter C, which determines how much the model penalizes margin violations. A smaller value of C allows a wider margin with more violations and usually leads to better generalization, while a larger C enforces stricter classification of training instances but increases the risk of overfitting.

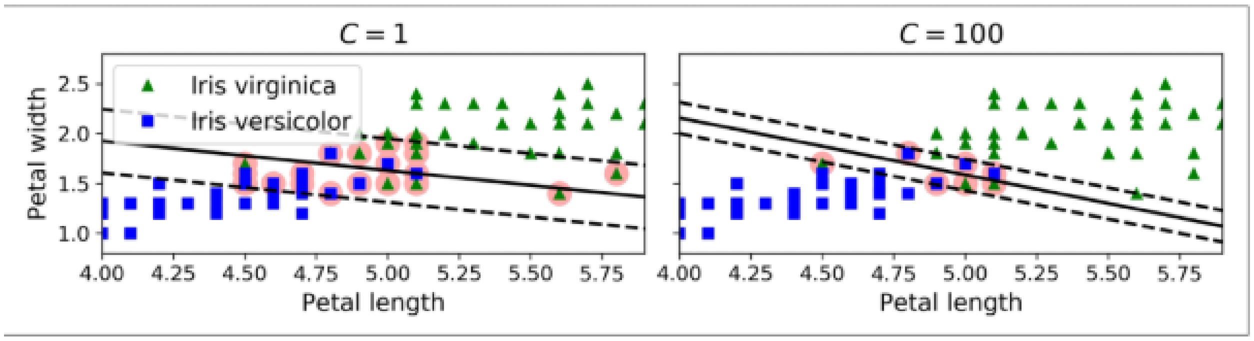


Figure 5-4. Large margin (left) versus fewer margin violations (right)

Although linear SVMs are effective in many scenarios, real-world data is often not linearly separable. One approach to handling **nonlinear** data is to add nonlinear features, such as polynomial features, that transform the data into a higher-dimensional space where a linear separator may exist. For example, data that is inseparable in one dimension may become separable after adding squared features.

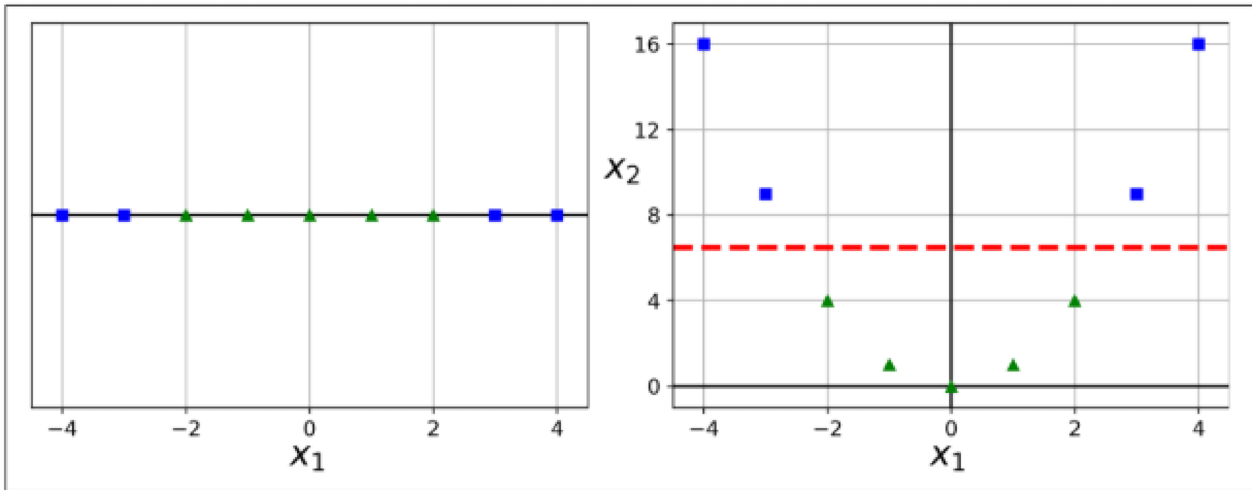


Figure 5-5. Adding features to make a dataset linearly separable

In practice, this idea can be implemented using a pipeline that applies polynomial feature expansion, feature scaling, and then a linear SVM classifier. This approach can successfully model nonlinear patterns, as demonstrated on datasets such as the “moons” dataset, but it becomes computationally expensive when the polynomial degree is high.

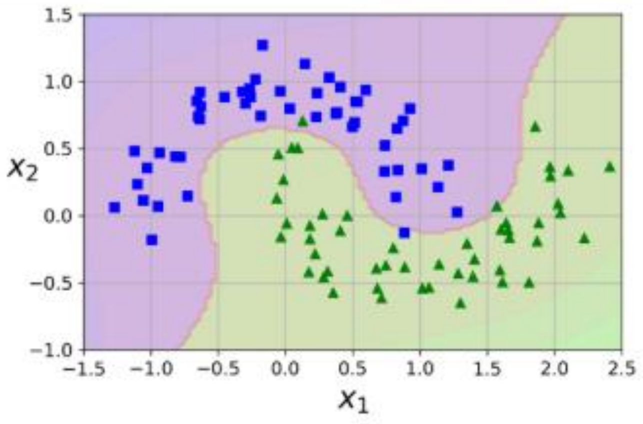


Figure 5-6. Linear SVM classifier using polynomial features

To address the inefficiency of explicit feature expansion, SVMs make use of the kernel trick, a mathematical technique that allows the model to behave as if it were operating in a high-dimensional feature space without actually computing the transformed features. **The polynomial kernel** enables SVMs to learn polynomial decision boundaries efficiently by computing inner products in the transformed space implicitly.

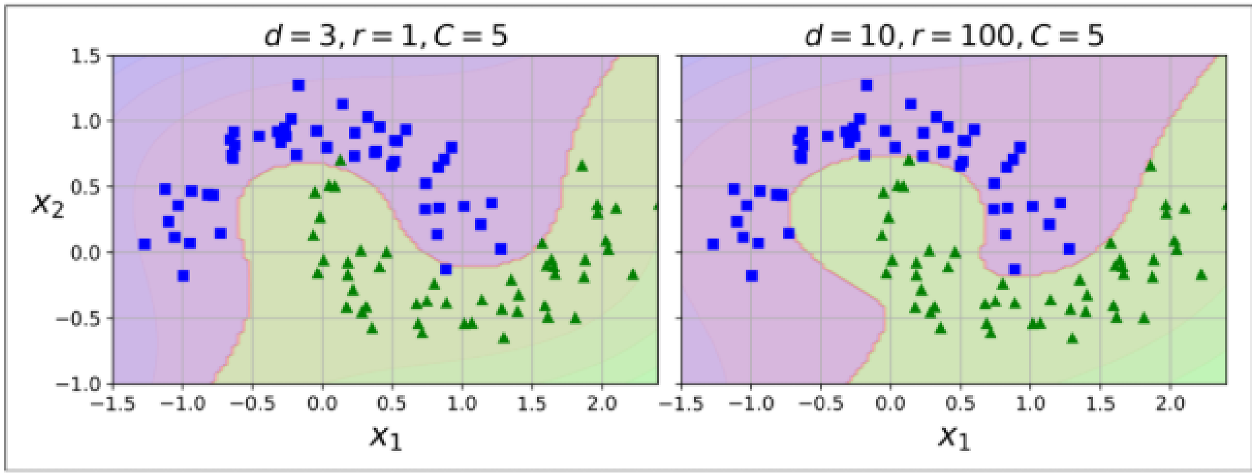


Figure 5-7. SVM classifiers with a polynomial kernel

Another intuitive way to handle nonlinear data is by constructing features based on similarity to landmarks. Using a Gaussian Radial Basis Function (RBF), each new feature measures how close a data point is to a specific landmark. This transformation often makes complex datasets linearly separable in the new feature space.

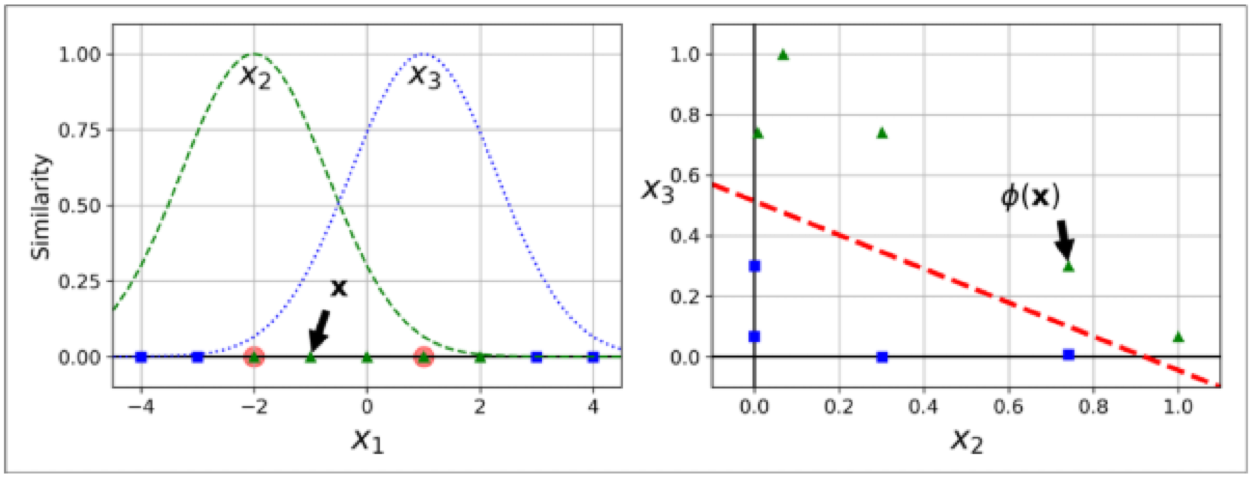


Figure 5-8. Similarity features using the Gaussian RBF

As with polynomial features, explicitly computing **similarity features** can be expensive. The **Gaussian RBF kernel** applies the kernel trick to achieve the same effect efficiently. Two hyperparameters control the behavior of this kernel: gamma (γ) and C. Gamma determines the radius of influence of individual training points, with larger values producing more complex and localized decision boundaries, while smaller values yield smoother boundaries. The C parameter continues to manage the trade-off between margin size and classification errors.

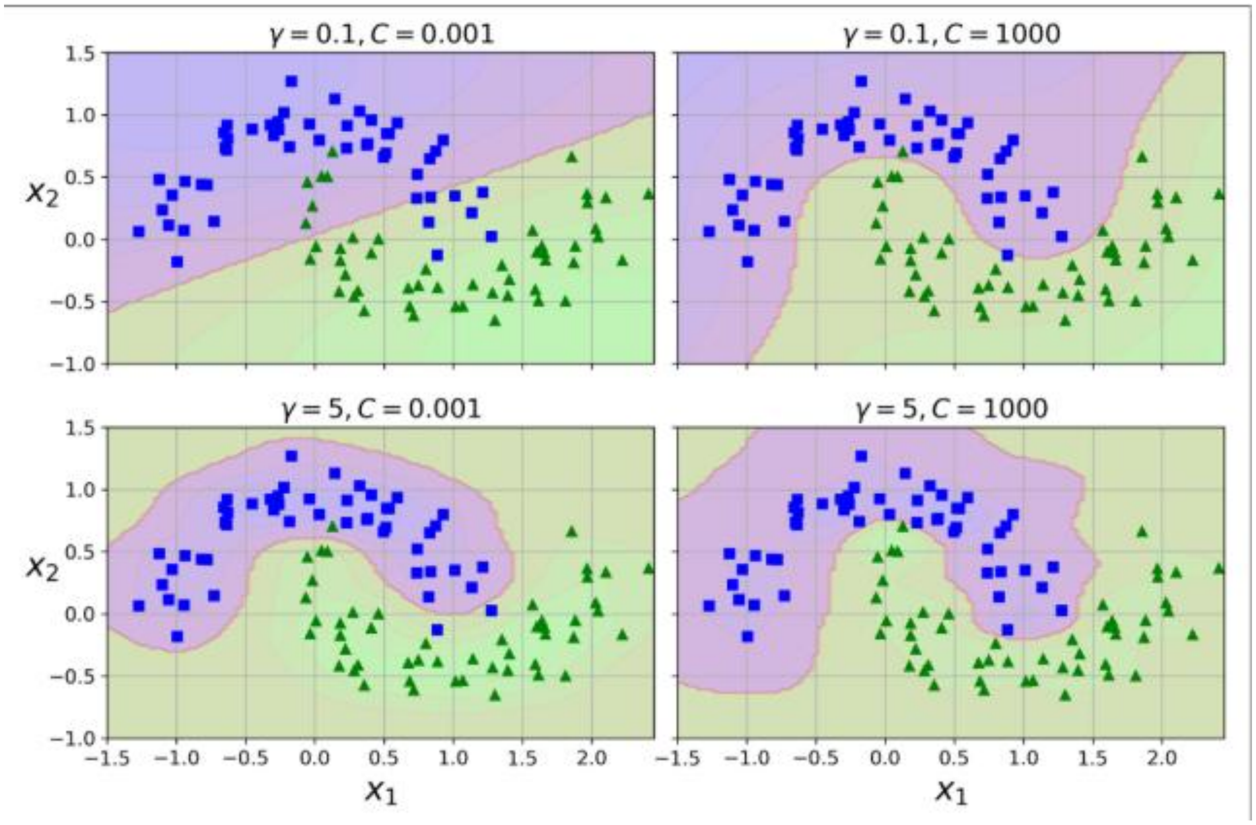


Figure 5-9. SVM classifiers using an RBF kernel

From a computational perspective, different SVM implementations are suited to different problem scales. LinearSVC is optimized for large datasets and scales approximately linearly with the number of training instances and features, but it does not support kernel methods. In contrast, SVC supports kernels but has much higher **computational complexity**, making it suitable primarily for small to medium-sized datasets.

Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

Table 5-1. Comparison of Scikit-Learn classes for SVM classification

SVMs are not limited to classification tasks; they can also be applied to regression**. SVM Regression** (SVR) reverses the classification objective by attempting to fit as many training instances as possible within a margin of width ε around the prediction function. Errors within this ε-tube are ignored, making the model ε-insensitive and robust to small deviations.

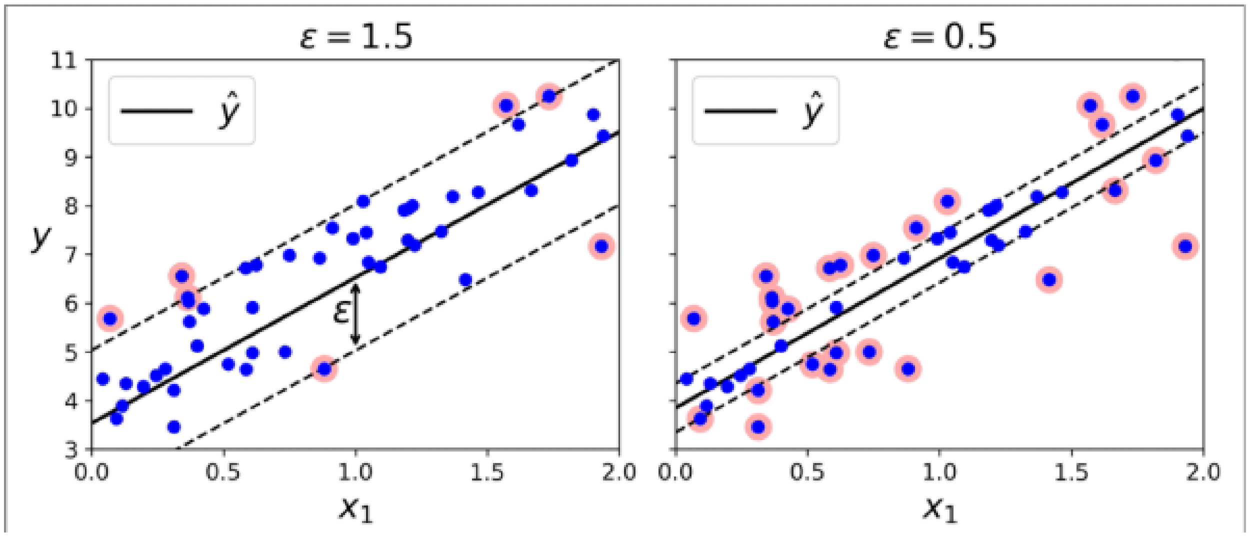


Figure 5-10. SVM Regression

For nonlinear regression problems, kernelized SVR can be used, such as SVR with a polynomial kernel. As with classification, the hyperparameter C controls the degree of regularization, influencing how closely the model fits the training data versus how smooth the resulting function is.

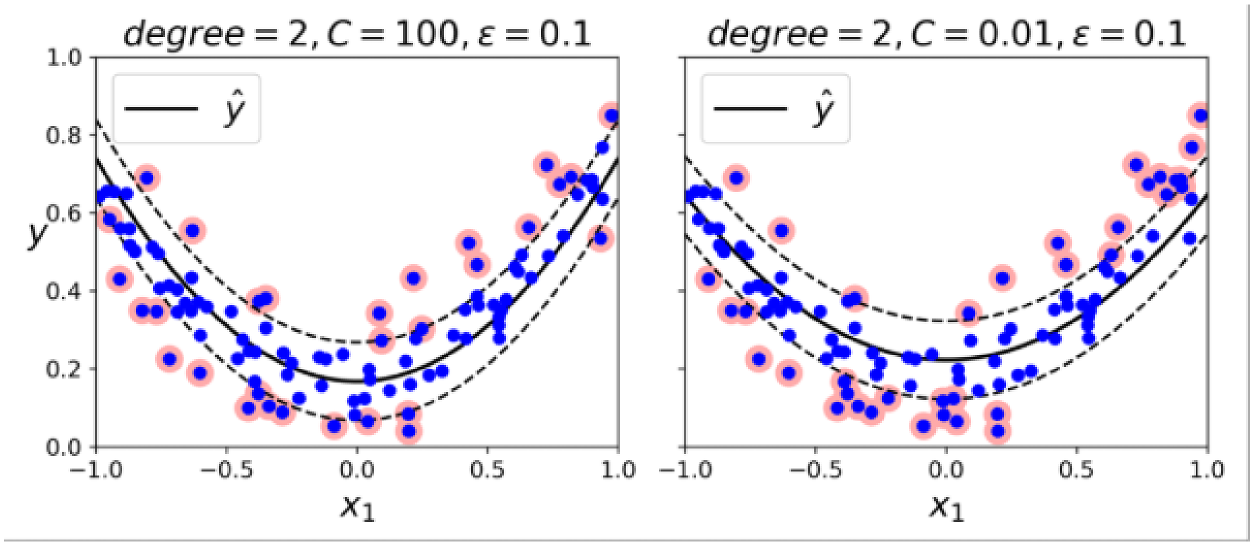


Figure 5-11. SVM Regression using a second-degree polynomial kernel

Internally, linear SVMs make predictions using a decision function of the form $w^T x + b$. The sign of this function determines the predicted class, and the decision boundary corresponds to the set of points where the function equals zero. The margin is inversely proportional to the norm of the weight vector $\| w \|$, meaning that smaller weights produce larger margins.

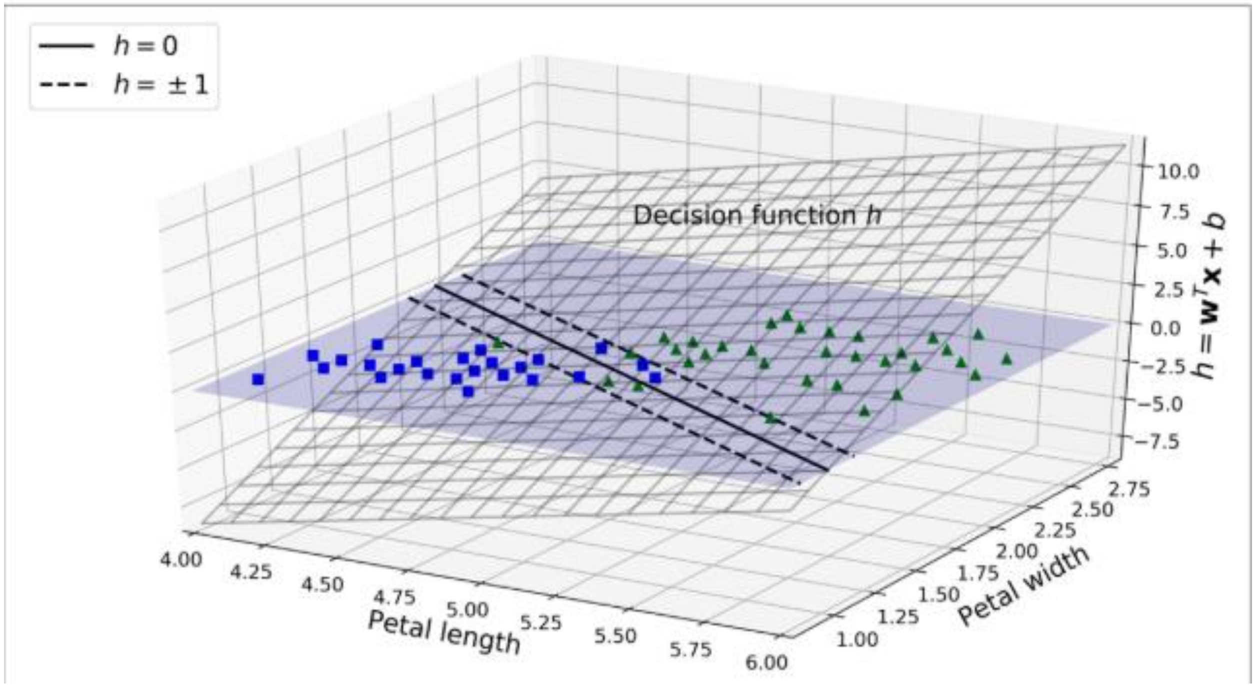


Figure 5-12. Decision function for the iris dataset

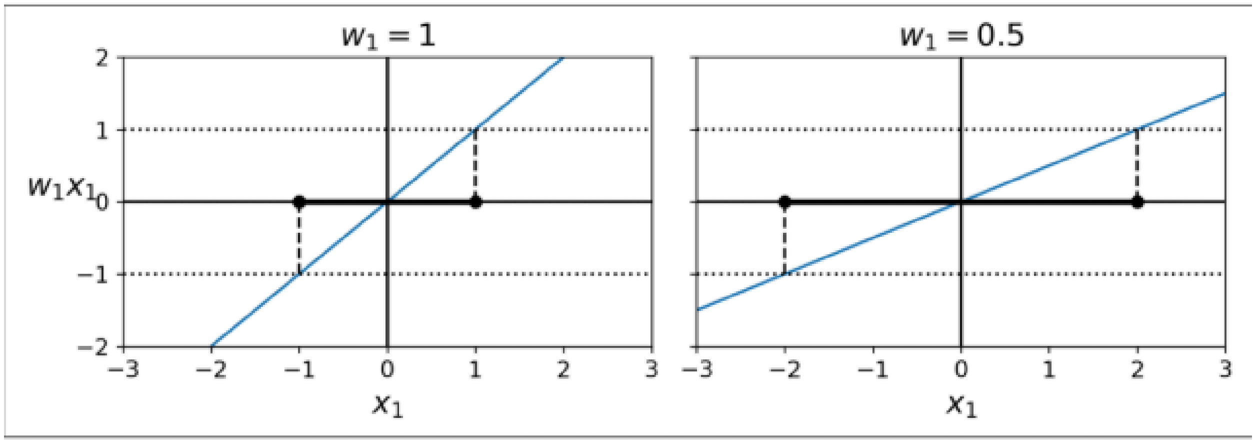


Figure 5-13. A smaller weight vector results in a larger margin

Training SVMs involves solving a convex optimization problem, often expressed as a quadratic programming problem. By reformulating this problem in its dual form, SVMs can leverage kernel functions to operate in very high-dimensional or even infinite-dimensional spaces efficiently. This dual formulation makes the kernel trick possible and ensures that predictions depend only on support vectors rather than all training instances.

Finally, SVMs can also be adapted for online learning. For linear SVMs, gradient-based optimization methods such as stochastic gradient descent can be used to update the model incrementally as new data arrives, although these approaches typically converge more slowly than **classical quadratic programming solutions**.