

Support Vector Machines

Concepts, Implementation, and
Applications

What is a Support Vector Machine?



Supervised learning model for classification and regression



Finds the optimal hyperplane that separates classes with the **maximum margin**



Effective in high-dimensional spaces



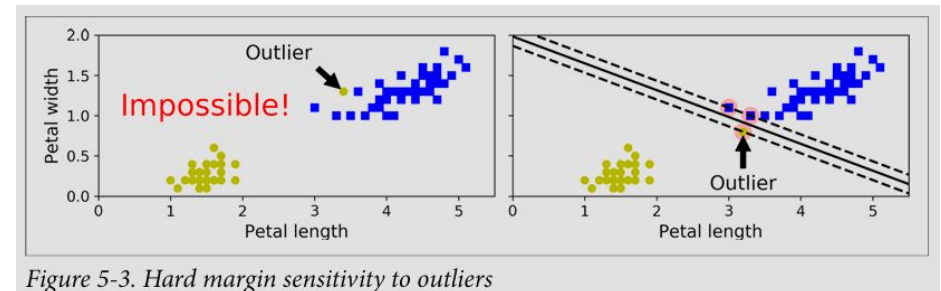
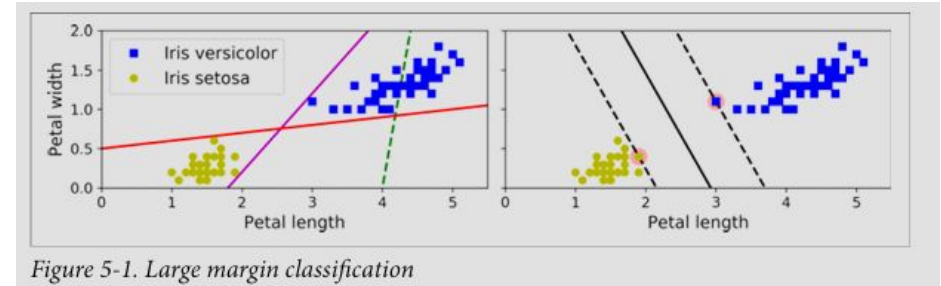
SVMs are well suited for classification of complex small- or medium-sized datasets



Robust to overfitting, especially in high-dimensional spaces

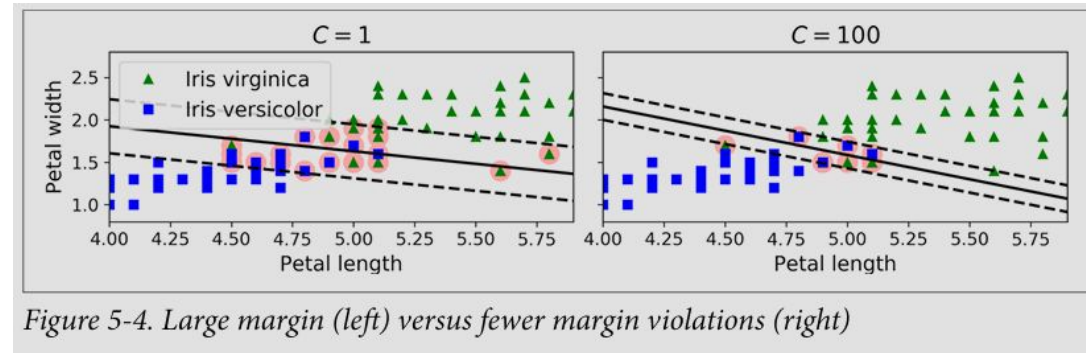
Linear SVM Classification

- **Hard Margin SVM:** Assumes data is linearly separable
- Maximizes the margin between classes
- Sensitive to outliers
- Instances located on the edge are called the support vectors.



Soft Margin Classification

- **Soft Margin SVM:** Handles non-separable data using slack variables (ξ)
- **Objective:** find a good balance between keeping the street as large as possible and limiting the margin violations.
- Controlled by hyperparameter C : trade-off between margin width and classification error



Nonlinear SVM Classification

- Use **kernel functions** to map data to higher dimensions
- Common kernels: Polynomial, Gaussian RBF, Sigmoid
- Enables SVM to handle nonlinear decision boundaries

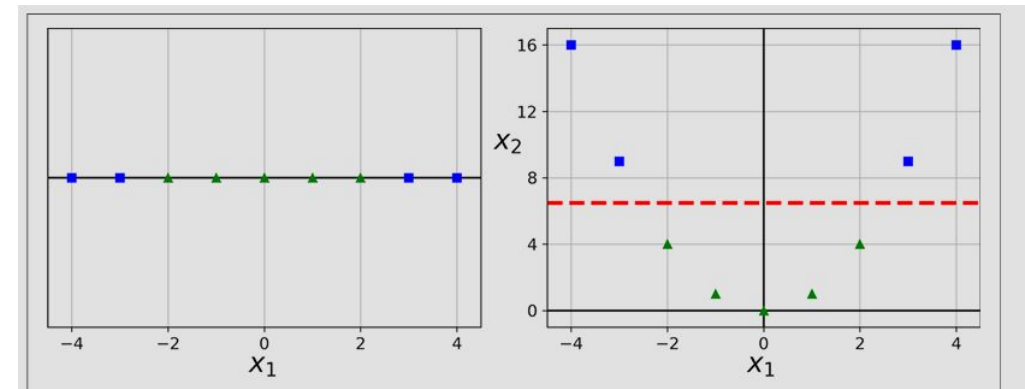
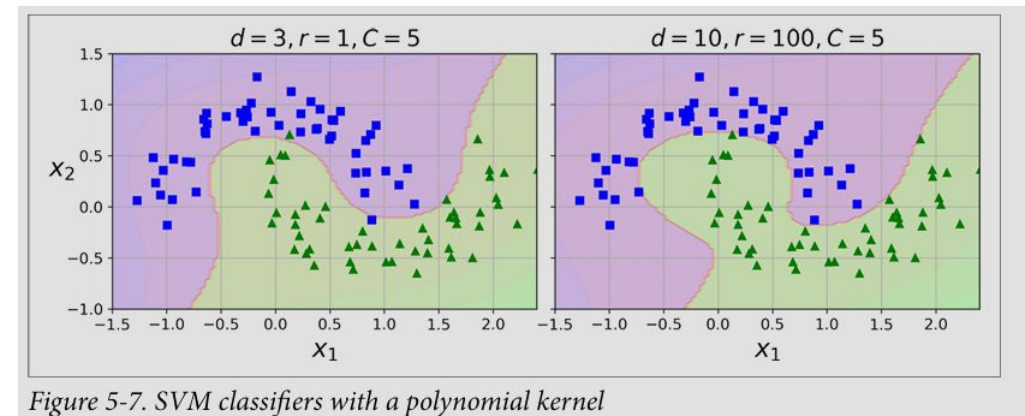


Figure 5-5. Adding features to make a dataset linearly separable

Polynomial Kernel

- Maps features into polynomial feature space
- At a low polynomial degree, model cannot deal with very complex datasets, and with a high polynomial degree it creates a huge number of features, making the model too slow.
- **The kernel trick:** get the same result as if you had added many polynomial features, even with very high-degree polynomials, without actually having to add them.



Similarity Features

- Tackle nonlinear problems
- Measures how much each instance resembles a particular landmark
- **Example**
 - Two landmarks at $x_1 = -2$ and $x_1 = 1$
 - The instance $x_1 = -1$: it is located at a distance of 1 from the first landmark and 2 from the second landmark. Therefore, its new features are $x_2 = \exp(-0.3 \times 1^2) \approx 0.74$ and $x_3 = \exp(-0.3 \times 2^2) \approx 0.30$.
- **Downside**
 - A training set with m instances and n features gets transformed into a training set with m instances and m features (dropping the original features).

Gaussian RBF

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

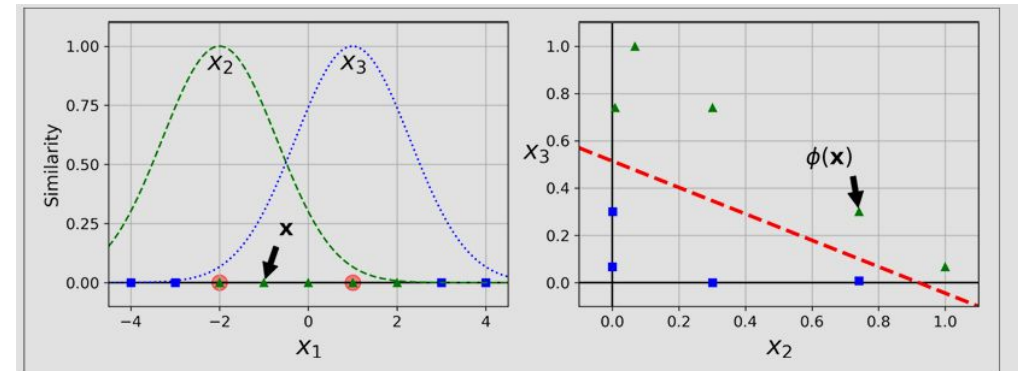


Figure 5-8. Similarity features using the Gaussian RBF

Gaussian RBF (Radial Basis Function) Kernel

- Most commonly used kernel
- Similarity-based feature mapping

Equation:

$$K(x, x') = \exp(-\gamma ||x - x'||^2)$$

- γ acts like a regularization hyperparameter:
 - Model overfitting → Reduce
 - Model underfitting → Increase

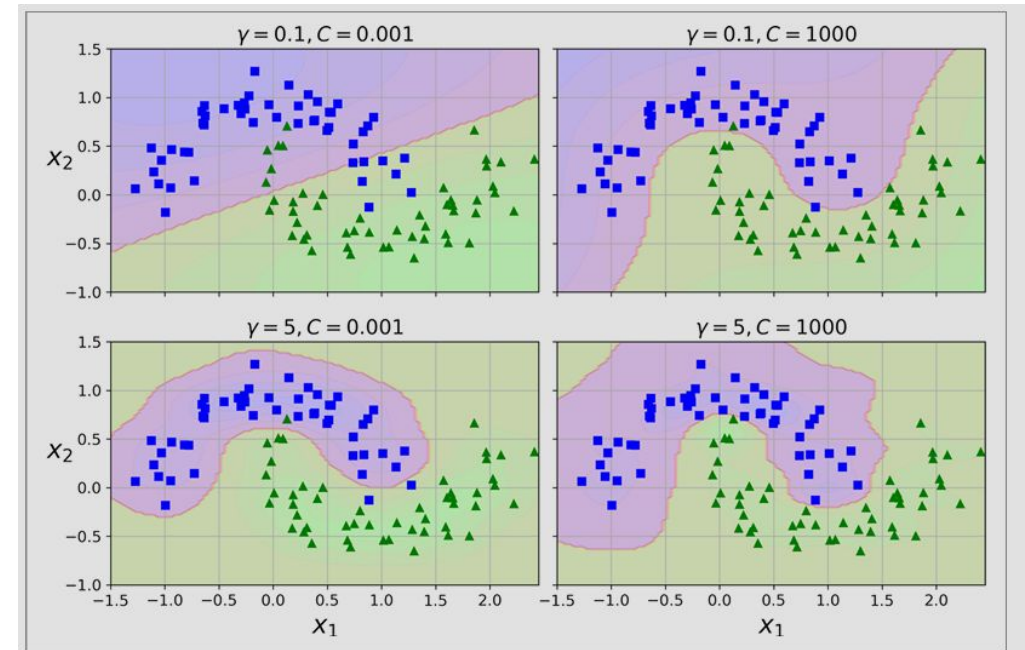


Figure 5-9. SVM classifiers using an RBF kernel

Computational Complexity

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

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

SVM Regression (SVR)

- SVM can also be used for regression
- The trick is to reverse the objective:
 - Instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM Regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances off the street).
- **Goal:** Fit as many instances as possible within a margin ϵ
- Controlled by ϵ and C

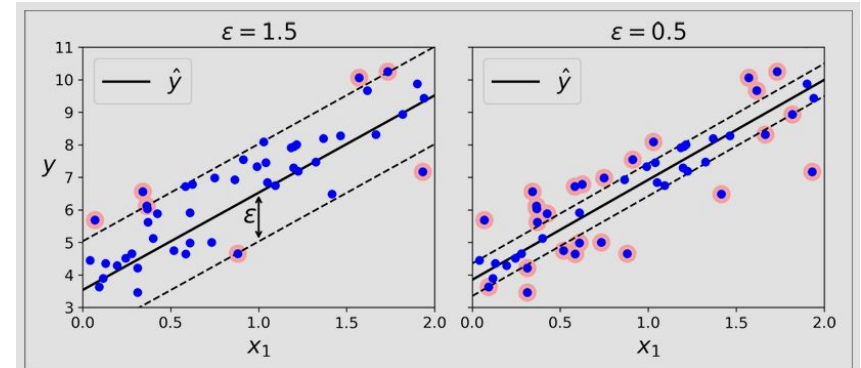


Figure 5-10. SVM Regression

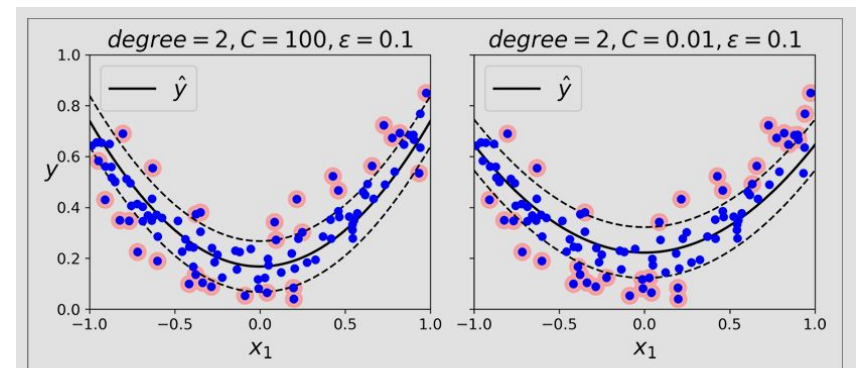
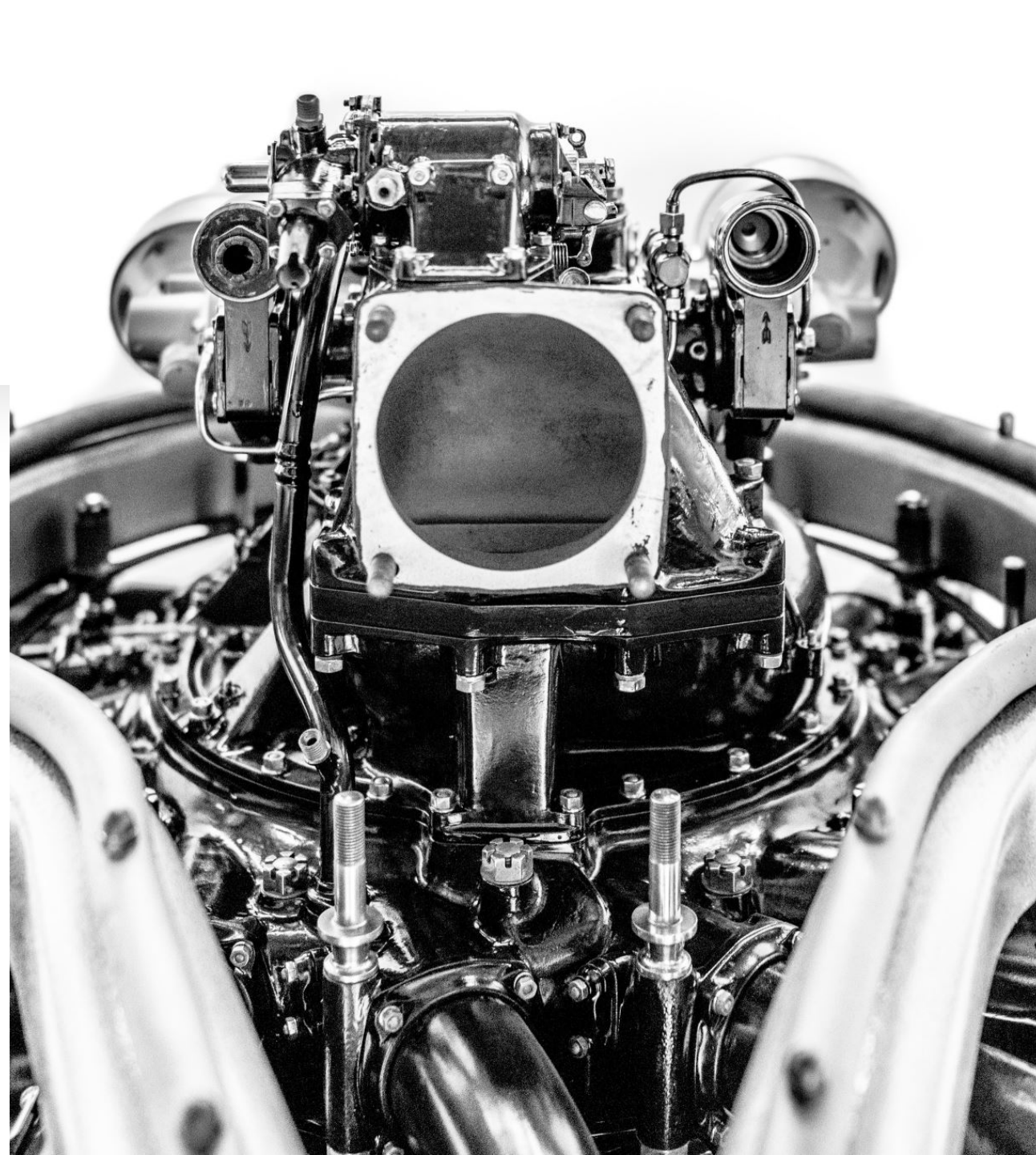


Figure 5-11. SVM Regression using a second-degree polynomial kernel

Under the Hood

-
- Explains how SVMs make predictions and how training algorithms work.
- The bias term will be called b , and the feature weights vector will be called w .
- No bias feature ($x_0 = 1$) will be added to the input feature vectors.



Decision Function and Predictions

- **Decision function**

$$w^T x + b = w_1 x_1 + \dots + w_n x_n + b.$$

- **Linear SVM classifier prediction**

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

- The decision boundary is the set of points where the decision function is equal to 0.
- The dashed lines represent the points where the decision function is equal to 1 or -1.
- Training a linear SVM classifier means finding the values of w and b that make this margin as wide as possible while avoiding margin violations (hard margin) or limiting them (soft margin).

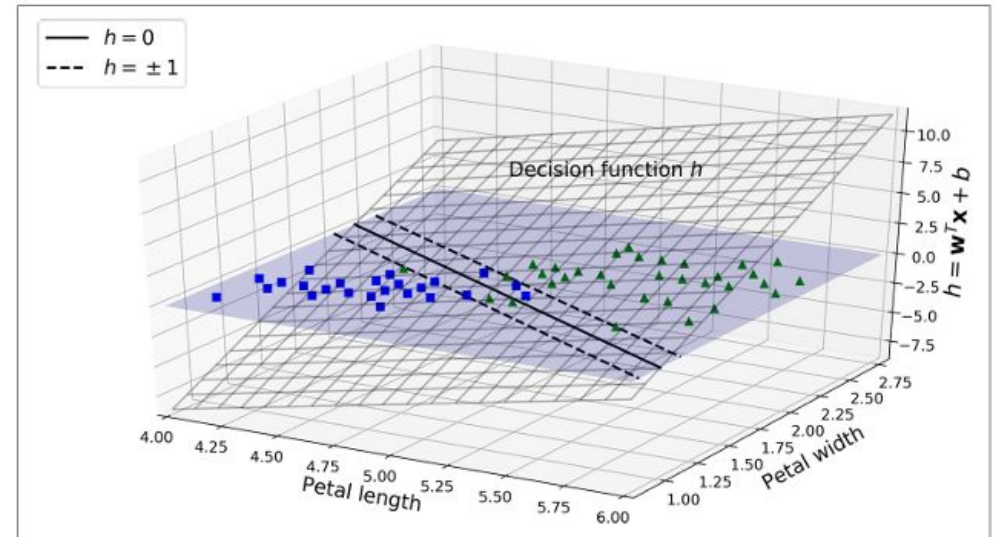


Figure 5-12. Decision function for the iris dataset

Training Objective

- The slope of the decision function is equal to the norm of the weight vector, $||w||$.
- Dividing the slope by 2 will multiply the margin by 2.

- **Hard margin linear SVM classifier objective**

$$\begin{aligned} & \underset{w, b}{\text{minimize}} \quad \frac{1}{2} w^T w \\ & \text{subject to } t^i (w^T x^i + b) \geq 1 \text{ for } i = 1, 2, \dots, m \end{aligned}$$

- Where $t^i = -1$ for negative instances (if $y^i = 0$) and $t^i = +1$ for positive instances (if $y^i = 1$)
- $||w||$ is not differentiable at $w = 0$.
Optimization algorithms work much better on differentiable functions.

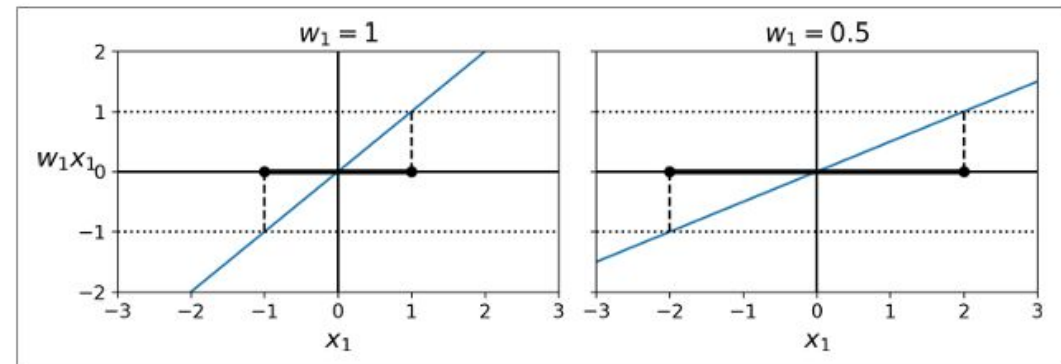


Figure 5-13. A smaller weight vector results in a larger margin

Training Objective

- **Soft margin linear SVM classifier Objective**

$$\begin{aligned} & \underset{w, b, \zeta}{\text{minimize}} \quad \frac{1}{2} w^T w + C \sum_{i=1}^m \zeta^i \\ & \text{subject to } t^i (w^T x^i + b) \geq 1 - \zeta^i \text{ and } \zeta^i \geq 0 \text{ for } i = 1, 2, \dots, m \end{aligned}$$

- Slack variable $\zeta(i) \geq 0$ measures how much the i th instance is allowed to violate the margin.
- **Two conflicting objectives:**
 - Make the slack variables as small as possible to reduce the margin violations
 - Make $\frac{1}{2} w^T w$ as small as possible to increase the margin
- The C hyperparameter allows us to define the trade off between these two objectives.

Kernelized SVMs

- **Second-degree polynomial mapping**
- The transformed vector is 3D instead of 2D

$$\phi(\mathbf{x}) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Kernelized SVMs

- **Kernel trick for a second-degree polynomial mapping**
- Apply second-degree polynomial mapping and compute the dot product of the transformed vectors
- The dot product of the transformed vectors is equal to the square of the dot product of the original vectors:
 $\phi(a)^T \phi(b) = (a^T b)^2$.

$$\begin{aligned}\phi(\mathbf{a})^T \phi(\mathbf{b}) &= \begin{pmatrix} a_1^2 \\ \sqrt{2} a_1 a_2 \\ a_2^2 \end{pmatrix}^T \begin{pmatrix} b_1^2 \\ \sqrt{2} b_1 b_2 \\ b_2^2 \end{pmatrix} = a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2 \\ &= (a_1 b_1 + a_2 b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 = (\mathbf{a}^T \mathbf{b})^2\end{aligned}$$

Kernelized SVMs

- The function $K(a, b) = (a^T b)^2$ is a second-degree polynomial kernel.
- In Machine Learning, a kernel is a function capable of computing the dot product $\phi(a)^T \phi(b)$, based only on the original vectors a and b , without having to compute the transformation ϕ .
- **Common kernels**
 - Linear: $K(a, b) = a^T b$
 - Polynomial: $K(a, b) = \gamma(a^T b + r)^d$
 - Gaussian RBF: $K(a, b) = \exp(-\gamma \|a - b\|^2)$
 - Sigmoid: $K(a, b) = \tanh(\gamma a^T b + r)$