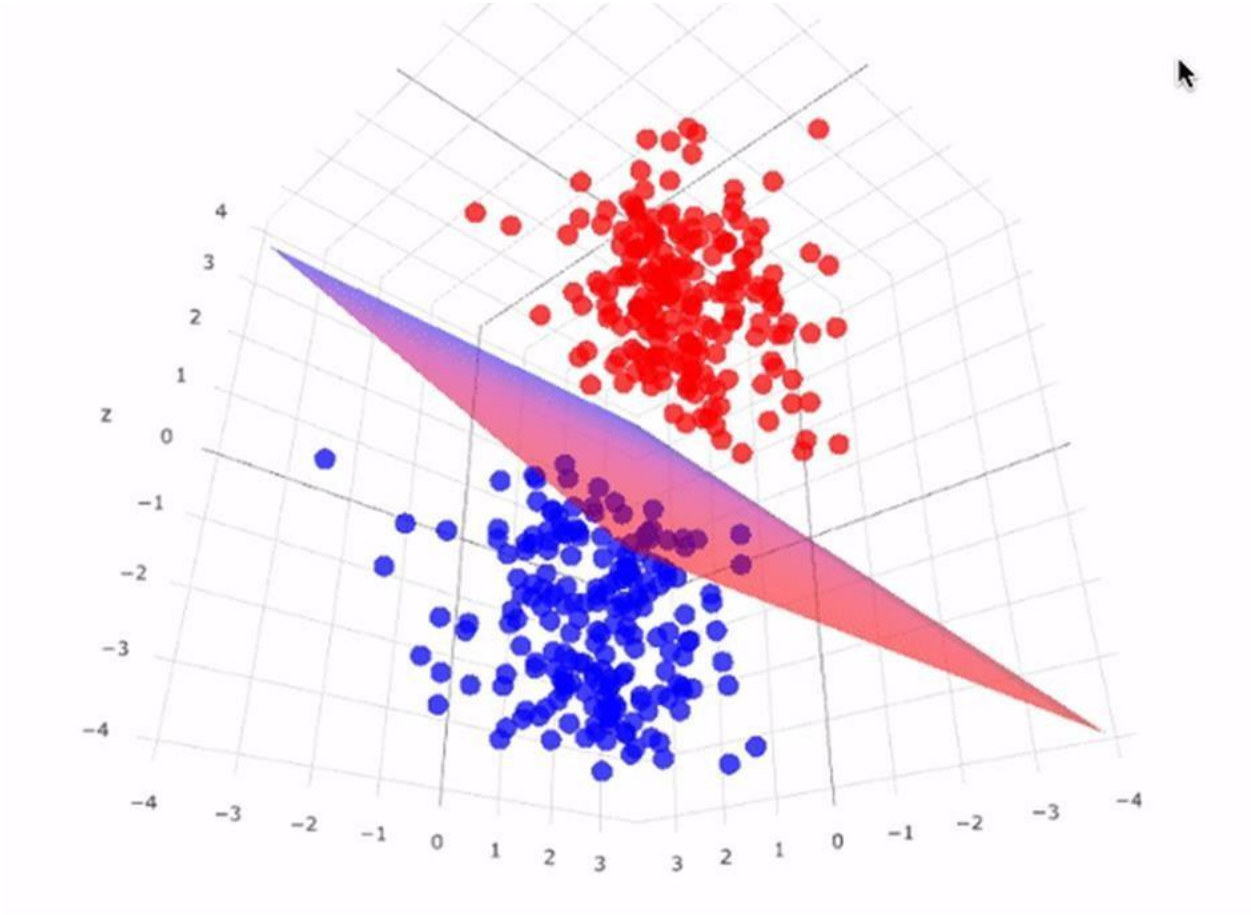
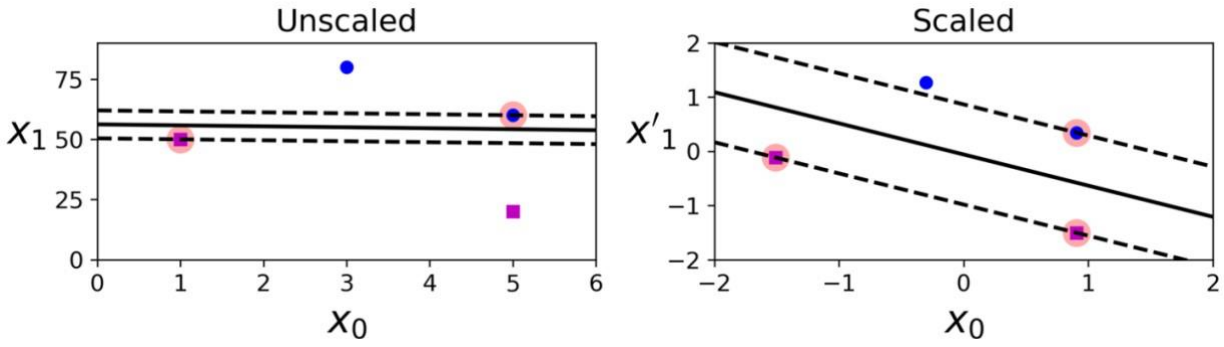
5. Support Vector Machines



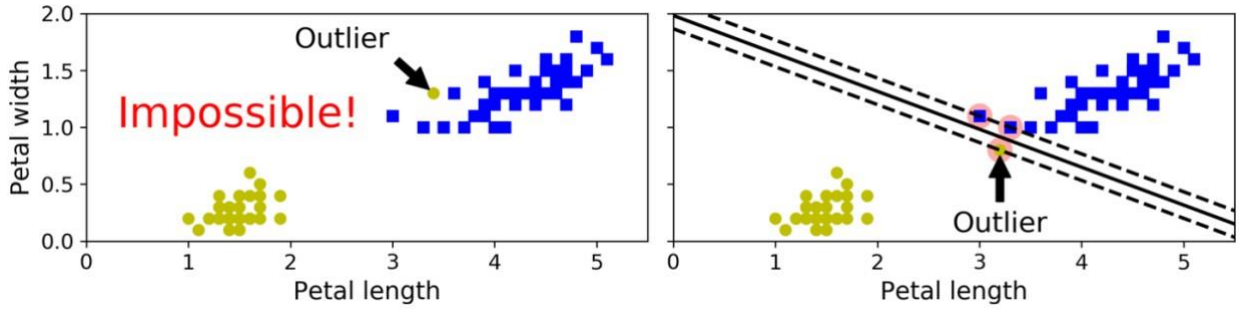
What are Support Vector Machines (SVMs)?

SVMs are a type of supervised learning algorithm primarily used for classification tasks, though they can also be used for regression. In supervised learning, you have a known set of inputs (features) and outputs (labels), and you want to train a model to learn the relationship between them.



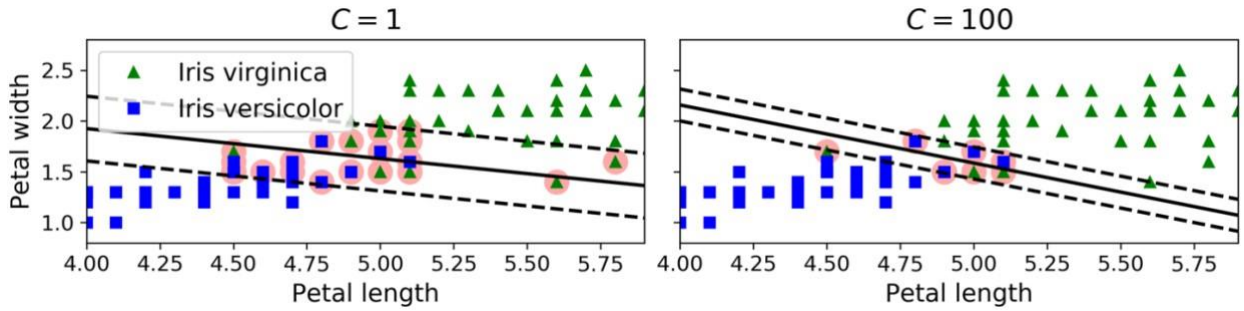
Soft Margin Classification

If we strictly impose that all instances must be off the street and on the right side, this is called hard margin classification. Two main issues with hard margin classification. First, it only works if the data is linearly separable. Second, it is sensitive to outliers. Also SVMs are sensitive to the feature scales.



To avoid these issues, use a more flexible model. The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations (i.e., instances that end up in the middle of the street or even on the wrong side). This is called soft margin classification.

When creating an SVM model using Scikit-Learn, we can specify a number of hyperparameters. C is one of those hyperparameters. If we set it to a low value, then we end up with the model on the left. With a high value, we get the model on the right. Margin violations are bad. It's usually better to have few of them. However, in this case the model on the left has a lot of margin violations but will probably generalize better.

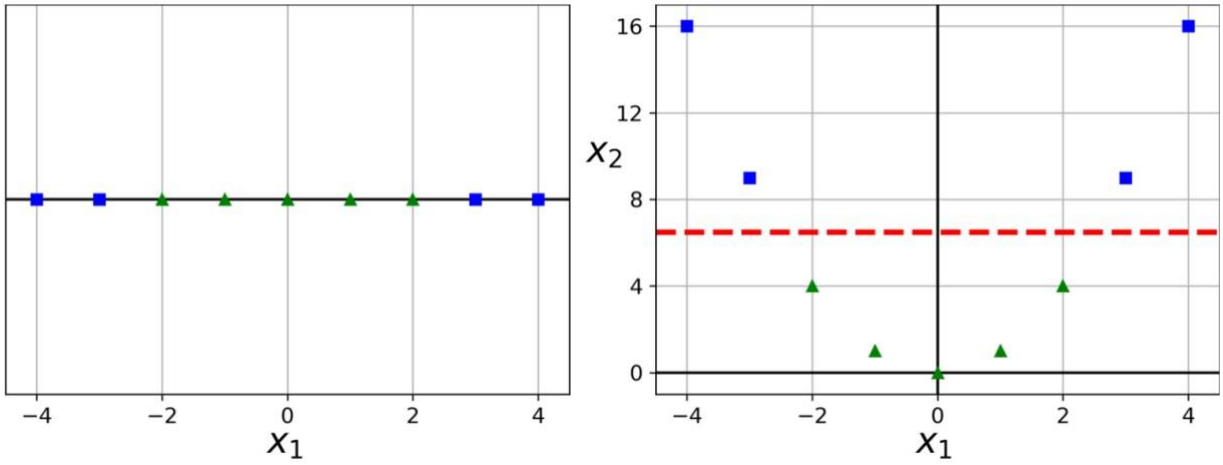


We could use Stochastic Gradient Descent to train a linear SVM classifier. It does not converge as fast as the LinearSVC class, but it can be useful to handle online classification tasks or huge datasets that do not fit in memory (out-of-core training).

Nonlinear SVM Classification

Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable. One approach to handling nonlinear datasets is to add more features, such as polynomial features in some cases this can result in a linearly separable dataset.

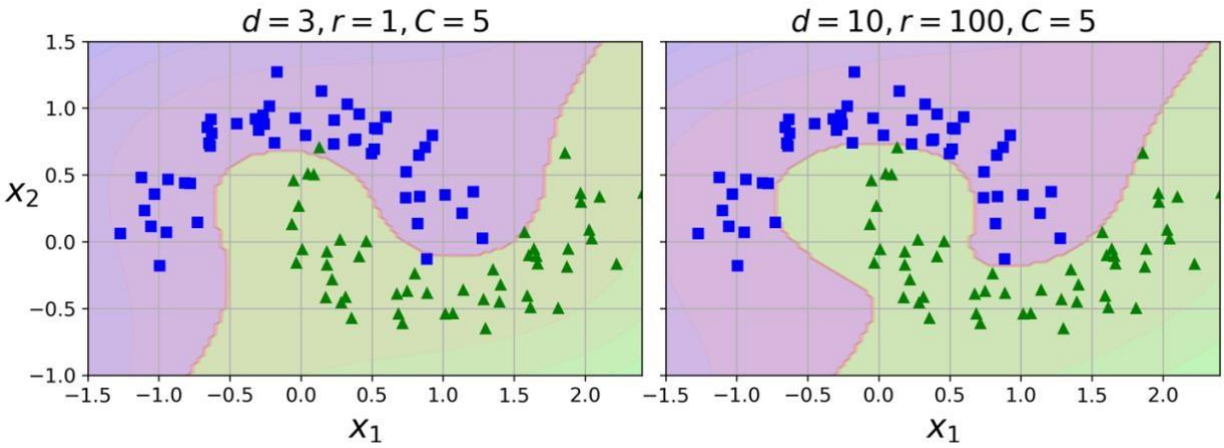
Therefore, when you are doing Feature engineering you create polynomial features, form linear ones.



Polynomial Kernel

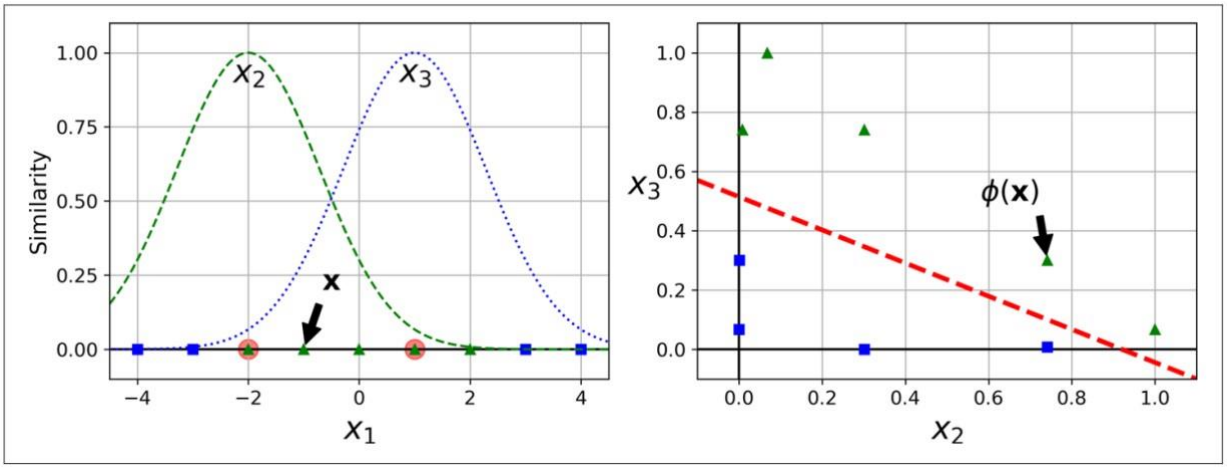
At a low polynomial degree, this method cannot deal with very complex datasets, and with a high polynomial degree it creates a huge number of features, making the model too slow.

You can use the kernel trick to tune the hyperparameters and see which parameters best fit the model. Grid search is used for finding the best one.



Similarity Features

A technique to tackle nonlinear problems is to add features computed using a similarity function, which measures how much each instance resembles a particular landmark. N features lead to n landmarks.



Gaussian RBF Kernel

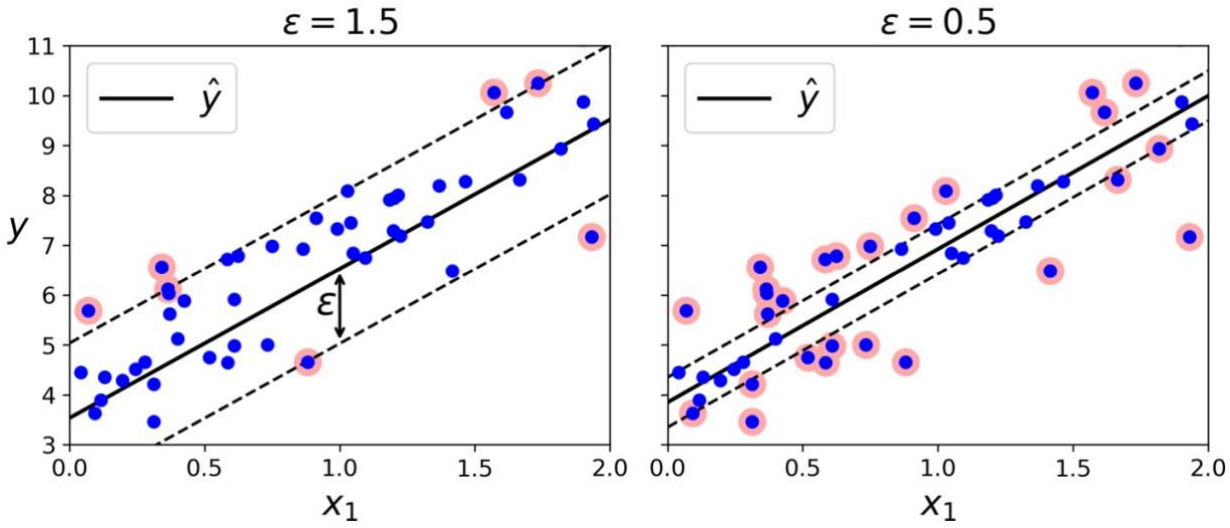
Just like the polynomial features method, the similarity features method can be useful with any Machine Learning algorithm, but it may be computationally expensive to compute all the additional features, especially on large training sets.

Computational Complexity

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 is well suited for small to middle datasets.

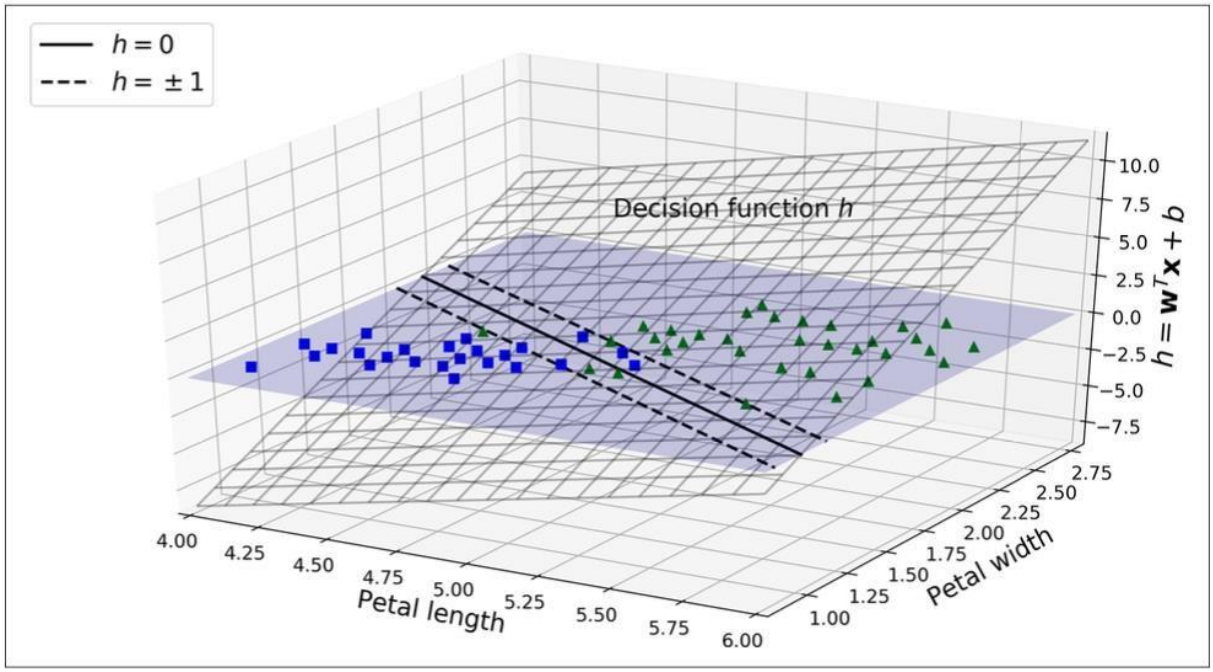
SVM Regression



It works the like classification, but now all the points must be within the street. Also, can be used for polynomials.

Training Objective of SVMs

- **Goal**: The aim is to minimize the weight vector ww to maximize the margin (the distance between the decision boundary and the nearest data points of each class).
- **Minimizing** ‖w‖‖w‖: By minimizing ‖w‖‖w‖, the margin is increased. This is visualized as a larger gap between the decision boundary and the nearest points in a 2D space.
- **Hard Margin**: For a hard margin (strict classification), the decision function must be greater than 1 for all positive instances and less than -1 for negative instances.



Soft Margin

- **Introduction of Slack Variables**: Slack variables ζ(i)ζ(i) are introduced for each instance to allow some margin violations. This creates a "soft margin".
- **Balancing Margin Width and Violations**: The objective is to minimize both the slack variables (to reduce margin violations) and the size of ww (to increase the margin). The hyperparameter C is used to balance these objectives.

Quadratic Programming (QP)

- **Solving SVM Problems**: Both hard and soft margin problems are types of QP problems, solvable using various algorithms.
- **QP Problem Formulation**: This involves minimizing a quadratic function subject to linear constraints.

$$
\begin{aligned}
\text{minimize} \quad & 3x_1^2 + 4x_2^2 + 2x_1x_2 + 5x_1 + x_2 \\
\text{subject to} \quad & x_1 \geq 0 \\
& x_2 \geq 0 \\
& x_1 + 2x_2 = 1
\end{aligned}
$$

The Dual Problem

- **Alternative to Primal Problem**: The dual problem is a closely related alternative to the primal problem (the original problem formulation). Solving the dual can be more efficient, especially with the kernel trick.

Kernelized SVMs

- **Kernel Trick**: This trick allows SVMs to operate in a high-dimensional space without explicitly transforming data into that space. It's computationally efficient.
- **Types of Kernels**: Common kernels include linear, polynomial, Gaussian RBF, and sigmoid kernels.

Making Predictions with Kernelized SVMs

- **Predictions Using Support Vectors**: Only support vectors (key data points) are used in making predictions.
- **Computing Bias**: A specific formula is used to compute the bias term in the presence of kernels.

Online SVMs

- **Incremental Learning**: Online SVM classifiers learn incrementally, ideal for situations where data arrives in a stream.
- **Gradient Descent**: It's used for minimizing the cost function in online linear SVM classifiers but converges slower than QP-based methods.
- **Hinge Loss Function**: Used in the cost function, it penalizes misclassified instances.

Why Use SVMs?

1. **Effective in High-Dimensional Spaces**: SVMs work well when you have a lot of features.
2. **Versatile**: Due to different kernel functions, SVMs can be adapted for various decision boundaries, from linear to complex non-linear separations.
3. **Accuracy**: They are known for providing accurate predictions.

Limitations

- **Speed and Size**: SVMs can be slower and require more memory for large datasets.
- **Kernel Choice**: Choosing the right kernel and parameters can be challenging without domain knowledge and experimentation.

Conclusion

SVMs are a powerful tool in the machine learning toolbox, especially useful for classification problems where the decision boundary between classes is not immediately obvious. By effectively using support vectors and maximizing margins, SVMs can achieve high accuracy in both linear and non-linear classification tasks.