

# 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 (as you did in [Chapter 4](#)); in some cases this can result in a linearly separable dataset. Consider the left plot in [Figure 5-5](#): it represents a simple dataset with just one feature  $x_1$ . This dataset is not linearly separable, as you can see. But if you add a second feature  $x_2 = (x_1)^2$ , the resulting 2D dataset is perfectly linearly separable.

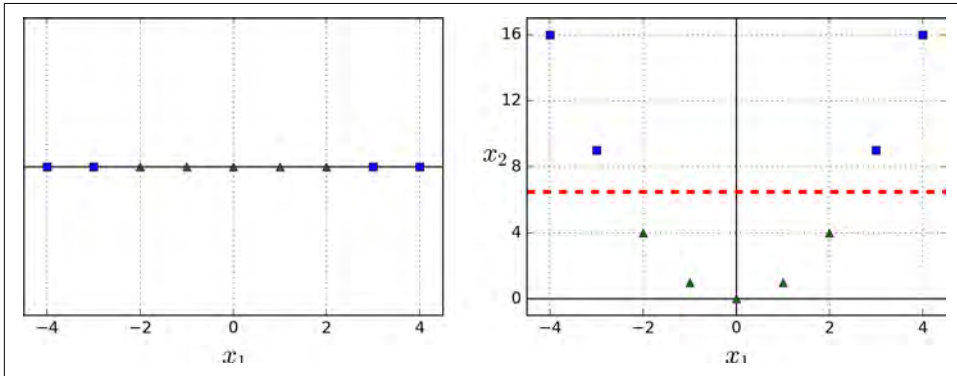


Figure 5-5. Adding features to make a dataset linearly separable

To implement this idea using Scikit-Learn, you can create a Pipeline containing a PolynomialFeatures transformer (discussed in “[Polynomial Regression](#)” on [page 121](#)), followed by a StandardScaler and a LinearSVC. Let’s test this on the moons dataset (see [Figure 5-6](#)):

```
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_svm_clf = Pipeline((
    ("poly_features", PolynomialFeatures(degree=3)),
    ("scaler", StandardScaler()),
    ("svm_clf", LinearSVC(C=10, loss="hinge"))
))

polynomial_svm_clf.fit(X, y)
```

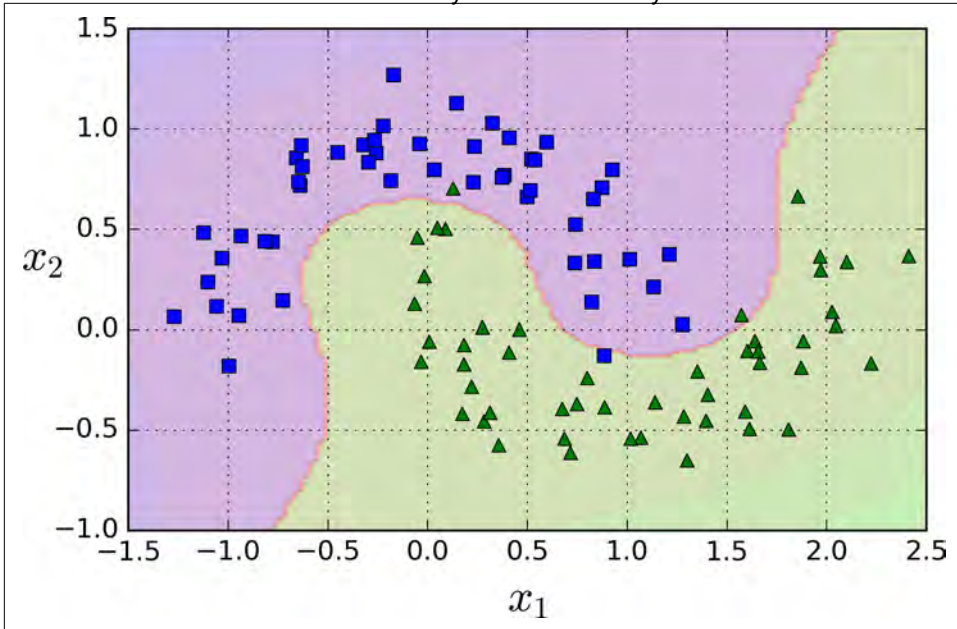


Figure 5-6. Linear SVM classifier using polynomial features

## Polynomial Kernel

Adding polynomial features is simple to implement and can work great with all sorts of Machine Learning algorithms (not just SVMs), but at a low polynomial degree it cannot deal with very complex datasets, and with a high polynomial degree it creates a huge number of features, making the model too slow.

Fortunately, when using SVMs you can apply an almost miraculous mathematical technique called the *kernel trick* (it is explained in a moment). It makes it possible to get the same result as if you added many polynomial features, even with very high-degree polynomials, without actually having to add them. So there is no combinatorial explosion of the number of features since you don't actually add any features. This trick is implemented by the SVC class. Let's test it on the moons dataset:

```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline((
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
))
poly_kernel_svm_clf.fit(X, y)
```

This code trains an SVM classifier using a 3<sup>rd</sup>-degree polynomial kernel. It is represented on the left of Figure 5-7. On the right is another SVM classifier using a 10<sup>th</sup>-degree polynomial kernel. Obviously, if your model is overfitting, you might want to