

reduce the polynomial degree. Conversely, if it is underfitting, you can try increasing it. The hyperparameter `coef0` controls how much the model is influenced by high-degree polynomials versus low-degree polynomials.

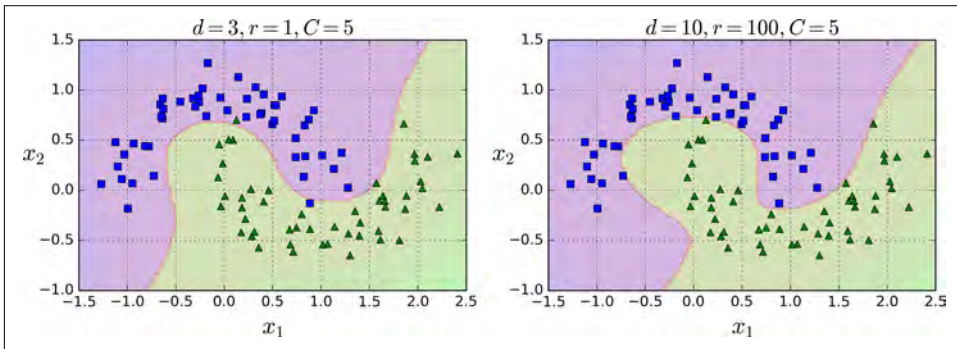


Figure 5-7. SVM classifiers with a polynomial kernel



A common approach to find the right hyperparameter values is to use grid search (see [Chapter 2](#)). It is often faster to first do a very coarse grid search, then a finer grid search around the best values found. Having a good sense of what each hyperparameter actually does can also help you search in the right part of the hyperparameter space.

## Adding Similarity Features

Another technique to tackle nonlinear problems is to add features computed using a *similarity function* that measures how much each instance resembles a particular *landmark*. For example, let's take the one-dimensional dataset discussed earlier and add two landmarks to it at  $x_1 = -2$  and  $x_1 = 1$  (see the left plot in [Figure 5-8](#)). Next, let's define the similarity function to be the Gaussian *Radial Basis Function* (RBF) with  $\gamma = 0.3$  (see [Equation 5-1](#)).

Equation 5-1. Gaussian RBF

$$\phi_{\gamma}(\mathbf{x}, \ell) = \exp(-\gamma \| \mathbf{x} - \ell \|^2)$$

It is a bell-shaped function varying from 0 (very far away from the landmark) to 1 (at the landmark). Now we are ready to compute the new features. For example, let's look at 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  $x_3 = \exp(-0.3 \times 2^2) \approx 0.30$ . The plot on the right of [Figure 5-8](#) shows the transformed dataset (dropping the original features). As you can see, it is now linearly separable.

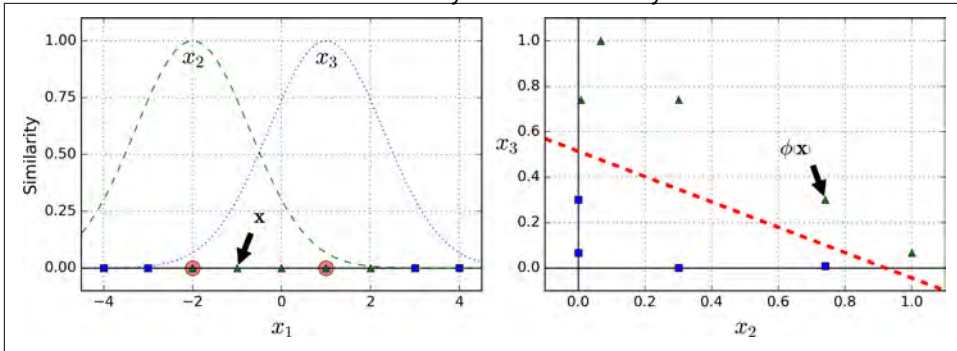


Figure 5-8. Similarity features using the Gaussian RBF

You may wonder how to select the landmarks. The simplest approach is to create a landmark at the location of each and every instance in the dataset. This creates many dimensions and thus increases the chances that the transformed training set will be linearly separable. The downside is that a training set with  $m$  instances and  $n$  features gets transformed into a training set with  $m$  instances and  $m$  features (assuming you drop the original features). If your training set is very large, you end up with an equally large number of features.

## 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. However, once again the kernel trick does its SVM magic: it makes it possible to obtain a similar result as if you had added many similarity features, without actually having to add them. Let's try the Gaussian RBF kernel using the SVC class:

```
rbf_kernel_svm_clf = Pipeline((
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
))
rbf_kernel_svm_clf.fit(X, y)
```

This model is represented on the bottom left of [Figure 5-9](#). The other plots show models trained with different values of hyperparameters  $\gamma$  and  $C$ . Increasing  $\gamma$  makes the bell-shape curve narrower (see the left plot of [Figure 5-8](#)), and as a result each instance's range of influence is smaller: the decision boundary ends up being more irregular, wiggling around individual instances. Conversely, a small  $\gamma$  value makes the bell-shaped curve wider, so instances have a larger range of influence, and the decision boundary ends up smoother. So  $\gamma$  acts like a regularization hyperparameter: if your model is overfitting, you should reduce it, and if it is underfitting, you should increase it (similar to the  $C$  hyperparameter).