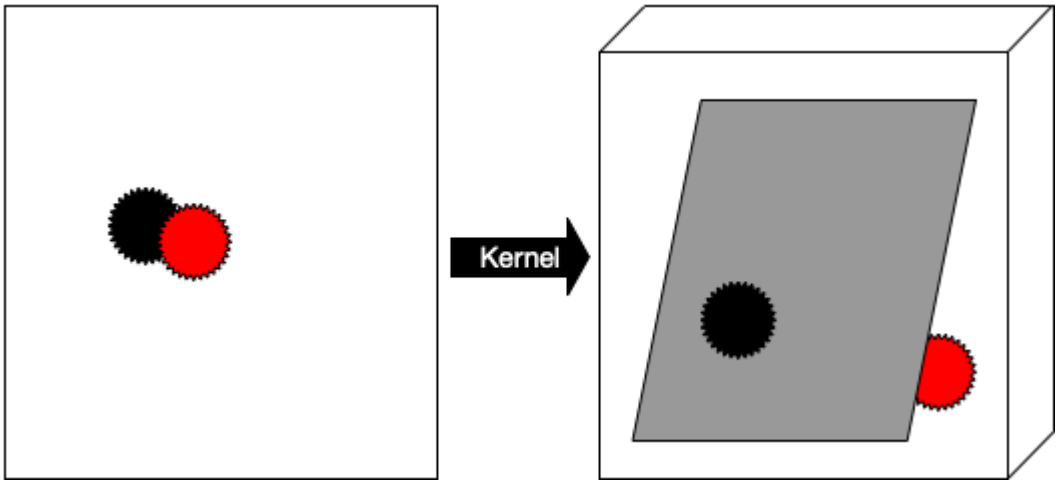

Figure 1.10 From 2D to 3D with a kernel.

A support vector machine is a binary classifier that implicitly maps data in feature space to higher dimensions in which data becomes separable by a linear plane, called a *hyperplane*. This mapping is implicit, and is carried out by a *kernel function*. This is a function that transforms the original input space to an alternative representation that implicitly has a higher dimensionality, with the aim of disentangling the data and making it linearly separable.

But the migration is implicit in the sense that it takes the form of a similarity function (φ in the picture above) applied to two feature vectors, just computing their distance. This is cordially called the *kernel trick*, It sounds like sheer magic, but it is actually quite simple. Let's take a look.

You should already be familiar with the dot product of two vectors. If not, please see Appendix 2 for a refresher. To recap, the standard dot product of two vectors *a* and *b* is the sum of the cross-product of the two vectors:

Listing 1.2 Dot product in Python.

```
def dot_product(a,b):
  return sum( [a[i]*b[i] for i in range(len(a))])
```

So, a dot product is just a multiplicative operation on two vectors that produces a single number.

Kernels are generalizations of this dot product between vectors: they compute the dot product between *altered* versions of these vectors. The nature of alteration is specified by a *kernel function* φ . Generally speaking, a kernel function takes two vectors, mixes in a constant (a kernel parameter) and adds some kernel-specific ingredients to produce a specific form of a dot product of the two vectors.

Let's return to our orange and apple. The objects are described by pairs of coordinates (x,y), since the table they're lying on is a flat XY-plane. Like othertypes of kernels, the so-called *polynomial kernel* maps lower-dimensional spaces to higher-dimensional ones. You may recall