guaranteed to converge efficiently. This is possible because we consider ϕ fixed and optimize only $\boldsymbol{\alpha}$, i.e., the optimization algorithm can view the decision function as being linear in a different space. Second, the kernel function k often admits an implementation that is significantly more computational efficient than naively constructing two $\phi(\boldsymbol{x})$ vectors and explicitly taking their dot product.

In some cases, $\phi(\boldsymbol{x})$ can even be infinite dimensional, which would result in an infinite computational cost for the naive, explicit approach. In many cases, $k(\boldsymbol{x}, \boldsymbol{x}')$ is a nonlinear, tractable function of $\boldsymbol{x}$ even when $\phi(\boldsymbol{x})$ is intractable. As an example of an infinite-dimensional feature space with a tractable kernel, we construct a feature mapping $\phi(x)$ over the non-negative integers x. Suppose that this mapping returns a vector containing x ones followed by infinitely many zeros. We can write a kernel function $k(x, x^{(i)}) = \min(x, x^{(i)})$ that is exactly equivalent to the corresponding infinite-dimensional dot product.

The most commonly used kernel is the **Gaussian kernel**

$$k(\boldsymbol{u}, \boldsymbol{v}) = \mathcal{N}(\boldsymbol{u} - \boldsymbol{v}; 0, \sigma^2 \boldsymbol{I}) \tag{5.84}$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the standard normal density. This kernel is also known as the **radial basis function** (RBF) kernel, because its value decreases along lines in $\boldsymbol{v}$ space radiating outward from $\boldsymbol{u}$. The Gaussian kernel corresponds to a dot product in an infinite-dimensional space, but the derivation of this space is less straightforward than in our example of the min kernel over the integers.

We can think of the Gaussian kernel as performing a kind of **template matching**. A training example $\boldsymbol{x}$ associated with training label y becomes a template for class y. When a test point $\boldsymbol{x}'$ is near $\boldsymbol{x}$ according to Euclidean distance, the Gaussian kernel has a large response, indicating that $\boldsymbol{x}'$ is very similar to the $\boldsymbol{x}$ template. The model then puts a large weight on the associated training label y. Overall, the prediction will combine many such training labels weighted by the similarity of the corresponding training examples.

Support vector machines are not the only algorithm that can be enhanced using the kernel trick. Many other linear models can be enhanced in this way. The category of algorithms that employ the kernel trick is known as **kernel machines** or **kernel methods** (Williams and Rasmussen, 1996; Schölkopf *et al.*, 1999).

A major drawback to kernel machines is that the cost of evaluating the decision function is linear in the number of training examples, because the i-th example contributes a term $\alpha_i k(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ to the decision function. Support vector machines are able to mitigate this by learning an $\boldsymbol{\alpha}$ vector that contains mostly zeros. Classifying a new example then requires evaluating the kernel function only for the training examples that have non-zero α_i. These training examples are known