

Lecture Notes on the Kernel Trick

Laurenz Wiskott*

first version 4 April 2006

last revision 12 May 2006

Contents

1 The kernel trick	1
1.1 Nonlinear input-output functions	1
1.2 Inner product in feature space and the kernel function	2
1.3 Input-output function in terms of the kernel function	3
1.4 Selecting the basis for the weight vector	3
1.5 Mercer's theorem	3

1 The kernel trick

1.1 Nonlinear input-output functions

If \mathbf{x} denotes an input vector and $\phi_k(\mathbf{x})$ with $k = 1, \dots, K$ is a set of fixed nonlinear scalar functions in \mathbf{x} , then

$$\bullet \quad g(\mathbf{x}) := \sum_k w_k \phi_k(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) \quad (1)$$

with real coefficients w_k **defines a nonlinear input-output function** $g(\mathbf{x})$. On the very right I have used the more compact vector notation with $\mathbf{w} := (w_1, \dots, w_K)^T$ and $\Phi := (\phi_1, \dots, \phi_K)^T$. Note, that not only \mathbf{x} but also the $\phi_k(\mathbf{x})$ can be regarded as vectors, which span a vector space of all functions that can be expressed in the form of equation (1). This vector space is often referred to as *function space* (D: Funktionenraum) and it has dimensionality K . Thus, Φ is a vector of vectors. For any concrete input vector \mathbf{x} , $\Phi(\mathbf{x})$ **can be considered a simple vector in a K -dimensional** Euclidean vector space, which is often referred to as *feature space* (D: Merkmalsraum). The process of going from \mathbf{x} to $\Phi(\mathbf{x})$ is referred to as a *mapping* (D: Abbildung) from input space to feature space or as a *nonlinear expansion* (D: nichtlineare Erweiterung) (\rightarrow Wiskott, 2006).

The dimensionality K of the feature space is usually much greater than the dimensionality of the input space. However, the manifold populated by expanded input vectors cannot be higher-dimensional than the input-space. For instance, if the input space is two-dimensional and feature space is three-dimensional, the two-dimensional manifold of input space gets embedded in

© 2006 Laurenz Wiskott

*Currently at the Institute for Theoretical Biology at Humboldt-University Berlin and the Bernstein Center for Computational Neuroscience Berlin, <http://itb.biologie.hu-berlin.de/~wiskott/>.

the three-dimensional feature space. It can be folded and distorted, but you cannot make it truly three-dimensional. Locally it will always be two-dimensional (think of a towel that is rolled up).

For high-dimensional feature spaces, the **mapping into and computations within feature space can become computationally very expensive**. However, some inner products in particular feature spaces can be computed very efficiently without the need to actually do the explicit mapping of the input vectors \mathbf{x} into feature space.

1.2 Inner product in feature space and the kernel function

As an example consider two input vectors \mathbf{x} and $\hat{\mathbf{x}}$ and the feature space of polynomials of degree two with the basis functions

$$\bullet \quad \phi_1 := 1, \tag{2}$$

$$\bullet \quad \phi_2 := \sqrt{2}x_1, \quad \phi_3 := \sqrt{2}x_2, \tag{3}$$

$$\bullet \quad \phi_4 := x_1^2, \quad \phi_5 := \sqrt{2}x_1x_2, \quad \phi_6 := x_2^2. \tag{4}$$

Now, if we take the Euclidean inner product of the input vectors mapped into feature space, we get

$$\bullet \quad \Phi(\mathbf{x})^T \Phi(\hat{\mathbf{x}}) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right) \left(1, \sqrt{2}\hat{x}_1, \sqrt{2}\hat{x}_2, \hat{x}_1^2, \sqrt{2}\hat{x}_1\hat{x}_2, \hat{x}_2^2\right)^T \tag{5}$$

$$\bullet \quad = 1^2 + 2x_1\hat{x}_1 + 2x_2\hat{x}_2 + x_1^2\hat{x}_1^2 + 2x_1x_2\hat{x}_1\hat{x}_2 + x_2^2\hat{x}_2^2 \tag{6}$$

$$\bullet \quad = (1 + x_1\hat{x}_1 + x_2\hat{x}_2)^2 \tag{7}$$

$$\bullet \quad = (1 + \mathbf{x}^T \hat{\mathbf{x}})^2. \tag{8}$$

(Note that one would get the same result if one would drop the factors $\sqrt{2}$ in (3) and (4) but would use a different inner product in feature space, namely $\mathbf{z} \cdot \hat{\mathbf{z}} := z_1\hat{z}_1 + 2z_2\hat{z}_2 + 2z_3\hat{z}_3 + z_4\hat{z}_4 + 2z_5\hat{z}_5 + z_6\hat{z}_6$. Thus, there is some arbitrariness here in the way the nonlinear expansion and the inner product in feature space are defined.)

Interestingly, **this generalizes to arbitrarily high exponents**, so that

$$\bullet \quad \Phi(\mathbf{x})^T \Phi(\hat{\mathbf{x}}) = (1 + \mathbf{x}^T \hat{\mathbf{x}})^d \tag{9}$$

is the inner product for a particular nonlinear expansion to polynomials of degree d . For $d = 3$ and

$$\phi_1 := 1, \tag{10}$$

$$\phi_2 := \sqrt{3}x_1, \quad \phi_3 := \sqrt{3}x_2, \tag{11}$$

$$\phi_4 := \sqrt{3}x_1^2, \quad \phi_5 := \sqrt{6}x_1x_2, \quad \phi_6 := \sqrt{3}x_2^2, \tag{12}$$

$$\phi_7 := x_1^3, \quad \phi_8 := \sqrt{3}x_1^2x_2, \quad \phi_9 := \sqrt{3}x_1x_2^2, \quad \phi_{10} := x_2^3, \tag{13}$$

we get

$$\Phi(\mathbf{x})^T \Phi(\hat{\mathbf{x}}) = (1 + \mathbf{x}^T \hat{\mathbf{x}})^3. \tag{14}$$

Equation (9) offers a very efficient way of computing the inner product in feature space, in particular for high degrees d . Because of the importance of the inner product between two expanded input vectors we give it a special name and **define the kernel function** (D: Kernfunktion)

$$\bullet \quad k(\mathbf{x}, \hat{\mathbf{x}}) := \Phi(\mathbf{x})^T \Phi(\hat{\mathbf{x}}). \tag{15}$$

Expressing everything in terms of inner products in feature space and using the kernel function to efficiently compute these inner products **is the kernel trick** (D: Kern-Trick). **It can**

save a lot of computation and permits us to use arbitrarily high-dimensional feature spaces. However, it requires that any computation in the feature space is formulated in terms of inner products, including the nonlinear input-output function, as will be discussed in the next section.

The kernel function given above is just one example, there exist several others (Müller et al., 2001).

1.3 Input-output function in terms of the kernel function

Since everything has to be expressed in terms of inner products in feature space in order to take advantage of the kernel trick, we have to rewrite (1). To do so we define \mathbf{w} as a linear combination of some expanded input vectors \mathbf{x}_j with $j = 1, \dots, J$. We then get

$$\bullet \quad \mathbf{w} := \sum_j \alpha_j \Phi(\mathbf{x}_j) \tag{16}$$

$$\bullet \quad g(\mathbf{x}) \stackrel{(1)}{=} \mathbf{w}^T \Phi(\mathbf{x}) \tag{17}$$

$$\bullet \quad \stackrel{(16)}{=} \sum_j \alpha_j \Phi(\mathbf{x}_j)^T \Phi(\mathbf{x}) \tag{18}$$

$$\bullet \quad \stackrel{(15)}{=} \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}). \tag{19}$$

The number J of input vectors \mathbf{x}_j used to define \mathbf{w} is usually taken to be much smaller than the dimensionality K of the feature space, because otherwise the computational complexity of computing $g(\mathbf{x})$ with the kernel trick (19) would be comparable to that of computing it directly (1) and there would be no advantage in using the kernel trick. This, however, defines a subspace of the feature space within which the optimization of $g(\mathbf{x})$ takes place and one is back again in not such a high-dimensional space. Thus, **the kernel trick allows you to work in a potentially very high-dimensional space, but you have to choose a moderate-dimensional subspace by selecting the basis for the weight vector before actually starting to optimize it.** Selecting this basis can be an art in itself.

1.4 Selecting the basis for the weight vector

There are three possibilities to choose the basis for the weight vectors.

- One simply takes all available data vectors \mathbf{x}^μ .
- One selects a subset of the data vectors \mathbf{x}^μ with a good heuristics that predicts their usefulness.
- One has a-priori ideas about what good basis vectors might be regardless of the data vectors.

1.5 Mercer's theorem

It is interesting to note that **since everything is done with the kernel function, it is not even necessary to know the feature space and the inner product within it.** It is only required that the kernel function is such that a corresponding feature space and inner product exists. There are criteria based on Mercer's theorem that guarantee this existence, see (Müller et al., 2001). **For instance,** it is obvious that **the kernel function must obey**

$$\bullet \quad k(\mathbf{x}, \hat{\mathbf{x}}) = k(\hat{\mathbf{x}}, \mathbf{x}) \tag{20}$$

$$\bullet \quad k(\mathbf{x}, \mathbf{x}) \geq 0 \quad \forall \mathbf{x} \tag{21}$$

otherwise it cannot induce a scalar product in feature space.

References

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12(2):181–201.

Wiskott, L. (2006). Lecture notes on nonlinear expansion. <http://itb.biologie.hu-berlin.de/~wiskott/Teaching/LectureNotes/NonlinearExpansion.pdf>.