Complex Decomposition of the Negative Distance Kernel

Tim vor der Brück
CC Distributed Secure Software Systems
Luzerne University of
Applied Sciences and Arts
tim.vorderbrueck@hslu.ch

Steffen Eger and Alexander Mehler
Text Technology Lab
Goethe University Frankfurt am Main
{steeger,amehler}@em.uni-frankfurt.de

***Abstract*—A *Support Vector Machine* (SVM) has become a very popular machine learning method for text classification. One reason for this relates to the range of existing kernels which allow for classifying data that is not linearly separable. The linear, polynomial and RBF (Gaussian *Radial Basis Function*) kernel are commonly used and serve as a basis of comparison in our study. We show how to derive the primal form of the quadratic *Power Kernel* (PK) – also called the *Negative Euclidean Distance Kernel* (NDK) – by means of complex numbers. We exemplify the NDK in the framework of text categorization using the *Dewey Document Classification* (DDC) as the target scheme. Our evaluation shows that the power kernel produces F-scores that are comparable to the reference kernels, but is – except for the linear kernel – faster to compute. Finally, we show how to extend the NDK-approach by including the Mahalanobis distance.**

***Keywords*—SVM, kernel function, text categorization**

INTRODUCTION

An SVM has become a very popular machine learning method for text classification [5]. One reason for its popularity relates to the availability of a wide range of kernels including the linear, polynomial and RBF (Gaussian radial basis function) kernel. This paper derives a decomposition of the quadratic *Power Kernel* (PK) using complex numbers and applies it in the area of text classification. Our evaluation shows that the NDK produces F-scores which are comparable to those produced by the latter reference kernels while being faster to compute – except for the linear kernel. This evaluation refers to the *Dewey Document Classification* DDC [11] as the target scheme and compares our NDK-based classifier with two competitors described in [6], [10] and [18], respecstively.

An SVM is a method for supervised classification introduced by [17]. It determines a hyperplane that allows for the best possible separation of the input data. (Training) data on the same side of the hyperplane is required to be mapped to the same class label. Furthermore, the margin of the hyperplane that is defined by the vectors located closest to the hyperplane is maximized. The vectors on the margin are called *Support Vectors* (SV). The decision function $dc : \mathbb{R}^n \rightarrow \{1, 0, -1\}$, which maps a vector to its predicted class, is given by: $dc(\mathbf{x}) := \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ where $\text{sgn} : \mathbb{R} \rightarrow \{1, 0, -1\}$ is the Signum function; the vector $\mathbf{w}$ and the constant b are determined by means of SV optimization. If $dc(\mathbf{x}) = 0$ then the vector is located directly on the hyperplane and no decision regarding both classes is possible.

The decision function is given in the primal form. It can be converted into the corresponding *dual form*: $dc(\mathbf{x}) = \text{sgn}(\sum_{j=1}^{m} y_j \alpha_j \langle \mathbf{x}, \mathbf{x}_j \rangle + b)$ where α_j and b are constants determined by the SV optimization and m is the number of SVs $\mathbf{x}_j$ (the input vector has to be compared only with these SVs). The vectors that are located on the wrong side of the hyperplane, that is, the vectors which prevent a perfect fit of SV optimization, are also considered as SVs. Thus, the number of SVs can be quite large. The scalar product, which is used to estimate the similarity of feature vectors, can be generalized to a kernel function. A kernel function K is a similarity function of two vectors such that the matrix of kernel values is symmetric and positive semidefinite. The kernel function only appears in the dual form. The decision function is given as follows: $dc(\mathbf{x}) := \text{sgn}(\sum_{j=1}^{m} y_j \alpha_j K(\mathbf{x}, \mathbf{x}_j) + b)$. Let $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^l$ be a function that transforms a vector into another vector mostly of higher dimensionality with $l \in \mathbb{N} \cup \{\infty\}$. It is chosen in such a way that the kernel function can be represented by: $K(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$. Thus, the decision function in the primal form is given by:

$$dc(\mathbf{x}) := \text{sgn}(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b) \quad (1)$$

Note that Φ is not necessarily uniquely defined. Furthermore, Φ might convert the data into a very high dimensional space. In such cases, the dual form should be used for the optimization process as well as for the classification of previously unseen data. One may think that the primal form is not needed. However, it has one advantage: if the normal vector of the hyperplane is known, a previously unseen vector can be classified just by applying the Signum function, Φ and a scalar multiplication. This is often much faster than computing the kernel function for previously unseen vectors and each SV as required when using the dual form.

The most popular kernel is the scalar product, also called the linear kernel. In this case, the transformation function Φ is the identity function: $K_{lin}(\mathbf{x}_1, \mathbf{x}_2) := \langle \mathbf{x}_1, \mathbf{x}_2 \rangle$. Another popular kernel function is the RBF (Gaussian Radial Basis Function), given by: $K_r(\mathbf{x}_1, \mathbf{x}_2) := e^{-\gamma ||\mathbf{x}_1 - \mathbf{x}_2||^2}$ where $\gamma \in \mathbb{R}$, $\gamma > 0$ is a constant that has to be manually specified. This kernel function can be represented by a function Φ_r that transforms the vector into infinite dimensional space [19]. For reasons of simplicity, assume that $\mathbf{x}$ is a vector with only one component. Then the transformation function Φ_r is given by:

$$\Phi_r(x) := e^{-\gamma x^2} [1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \ldots]^\top \quad (2)$$

Another often used kernel is the polynomial kernel:

$$K_p(\mathbf{x}_1, \mathbf{x}_2) := (a\langle \mathbf{x}_1, \mathbf{x}_2\rangle + c)^d \tag{3}$$

For $d := 2$, $a = 1$, $c = 1$, and two vector components, the function Φ_p is given by [9]: $\Phi_p : \mathbb{R}^2 \to \mathbb{R}^6$ with

$$\Phi_p(\mathbf{x}) = (1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2) \tag{4}$$

In general, a vector of dimension n is transformed by Φ_p into a vector of dimension $\binom{n+d}{d}$. Thus, the polynomial kernel leads to a large number of dimensions in the target space. However, the number of dimensions is not infinite as in the case of the RBF kernel.

THE POWER KERNEL

The PK is a conditionally positive definite kernel given by

$$K_s(\mathbf{x}_1, \mathbf{x}_2) := -||\mathbf{x}_1 - \mathbf{x}_2||^p \tag{5}$$

for some $p \in \mathbb{R}$ [16], [2], [12]. A kernel is called conditionally positive-definite if it is symmetric and satisfies the conditions [14]

$$\sum_{j,k=1}^{n} c_i\overline{c_j}K(\mathbf{x}_j, \mathbf{x}_k) \geq 0 \ \forall c_i \in \mathbb{K} \text{ with } \sum_{i=1}^{m} c_i = 0 \tag{6}$$

where $\overline{c_j}$ is the complex-conjugate of c_j. We consider here a generalized form of the PK for $p := 2$ (also called NDK):

$$K_{pow}(\mathbf{x}_1, \mathbf{x}_2) := -a||\mathbf{x}_1 - \mathbf{x}_2||^2 + c \tag{7}$$

with $a, c \in \mathbb{R}$ and $a > 0$. The expression $-a||\mathbf{x}_1 - \mathbf{x}_2||^2 + c$ can also be written as:

$$\begin{aligned} &- a\langle(\mathbf{x}_1 - \mathbf{x}_2), (\mathbf{x}_1 - \mathbf{x}_2)\rangle + c = \\ &- a(\langle \mathbf{x}_1, \mathbf{x}_1\rangle - 2\langle \mathbf{x}_1, \mathbf{x}_2\rangle + \langle \mathbf{x}_2, \mathbf{x}_2\rangle) + c \end{aligned} \tag{8}$$

For deciding which class a previously unseen vector belongs to we can use the decision function in the dual form:

$$\begin{aligned} dc(\mathbf{x}) :&= \operatorname{sgn}(\sum_{j=1}^{m} y_j\alpha_j K_{pow}(\mathbf{x}, \mathbf{x}_j) + b) \\ &= \operatorname{sgn}(\sum_{j=1}^{m} y_j\alpha_j(-a||\mathbf{x} - \mathbf{x}_j||^2 + c) + b) \end{aligned} \tag{9}$$

The decision function shown in formula (9) has the drawback that the previously unseen vector has to be compared with each SV, which can be quite time consuming. This can be avoided, if we reformulate formula (9) to:

$$\begin{aligned} dc(\mathbf{x}) =& \operatorname{sgn}(\sum_{j=1}^{m} y_j\alpha_j(-a\langle \mathbf{x}, \mathbf{x}\rangle + 2a\langle \mathbf{x}, \mathbf{x}_j\rangle - \\ & a\langle \mathbf{x}_j, \mathbf{x}_j\rangle + c) + b) \\ =& \operatorname{sgn}(-a\sum_{j=1}^{m} y_j\alpha_j\langle \mathbf{x}, \mathbf{x}\rangle + 2a\sum_{j=1}^{m} y_j\alpha_j\langle \mathbf{x}, \mathbf{x}_j\rangle + \\ & \sum_{j=1}^{m} y_j\alpha_j(-a\langle \mathbf{x}_j, \mathbf{x}_j\rangle + c) + b) \\ =& \operatorname{sgn}(-a\langle \mathbf{x}, \mathbf{x}\rangle\sum_{j=1}^{m} y_j\alpha_j + 2\langle \mathbf{x}, a\sum_{j=1}^{m} y_j\alpha_j\mathbf{x}_j\rangle - \\ & a\sum_{j=1}^{m} y_j\alpha_j\langle \mathbf{x}_j, \mathbf{x}_j\rangle + c\sum_{i=1}^{m} y_j\alpha_j + b) \end{aligned} \tag{10}$$

With $\mathbf{z} := a\sum_{j=1}^{m} y_j\alpha_j\mathbf{x}_j$, $\mathbf{u} := a\sum_{j=1}^{m} y_j\alpha_j\langle \mathbf{x}_j, \mathbf{x}_j\rangle$ and $c' = c\sum_{i=1}^{m} y_j\alpha_j$, formula (10) can be rewritten as:

$$dc(\mathbf{x}) = \operatorname{sgn}(-a\langle \mathbf{x}, \mathbf{x}\rangle\sum_{i=1}^{m} y_j\alpha_j + 2\langle \mathbf{x}, \mathbf{z}\rangle - \mathbf{u} + c' + b) \tag{11}$$

The expressions $\mathbf{u}$, $\mathbf{z}$, $(\sum_{i=1}^{m} y_j\alpha_j)$, and c' are identical for every vector $\mathbf{x}$ and can be precomputed. Note that there exists no primal form for the NDK based on real number vectors which is stated by the following proposition.

Proposition 1 Let a,$c \in \mathbb{R}$ with $a > 0$ and $n, l \in \mathbb{N}$. Then there is no function $\Phi_{re} : \mathbb{R}^n \to \mathbb{R}^l$ (re indicates that Φ_{re} operates on real numbers) with $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n : \langle \Phi_{re}(\mathbf{x}_1), \Phi_{re}(\mathbf{x}_2)\rangle = -a||\mathbf{x}_1 - \mathbf{x}_2||^2 + c$.

Proof: If such a function existed, then, for all $\mathbf{x} \in \mathbb{R}^n$,

$$||\Phi_{re}(\mathbf{x})||^2 = \langle \Phi_{re}(\mathbf{x}), \Phi_{re}(\mathbf{x})\rangle = -a \cdot 0 + c = c$$

which requires that $c \geq 0$ since the square of a real number cannot be negative. Now, consider $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $||\mathbf{x}-\mathbf{y}||^2 > \frac{2c}{a} \geq 0$. On the one hand, we have, by the Cauchy-Schwarz inequality,

$$|\langle \Phi_{re}(\mathbf{x}), \Phi_{re}(\mathbf{y})\rangle| \leq ||\Phi_{re}(\mathbf{x})|| \cdot ||\Phi_{re}(\mathbf{y})|| = \sqrt{c} \cdot \sqrt{c} = c.$$

On the other hand, it holds that

$$|-a||\mathbf{x} - \mathbf{y}||^2 + c| = |a||\mathbf{x} - \mathbf{y}||^2 - c| > 2c - c = c,$$

a contradiction. ■

Although no primal form and therefore no function Φ_{re} exists for real number vectors, such a function can be given if complex number vectors are used instead. In this case, the function Φ_c is defined with a real domain and a complex co-domain: $\Phi_c : \mathbb{R}^n \to \mathbb{C}^{4n+1}$ and

$$\begin{aligned} \Phi_c(\mathbf{x}) :=& (\sqrt{a}(x_1^2 - 1), \sqrt{a}i, \sqrt{2a}x_1, \sqrt{a}ix_1^2, \ldots, \\ & \sqrt{a}(x_n^2 - 1), \sqrt{a}i, \sqrt{2a}x_n, \sqrt{a}ix_n^2, \sqrt{c})^\top \end{aligned} \tag{12}$$

Note that no scalar product can be defined for complex numbers that fulfills the usual conditions of bilinearity and positive-definiteness simultaneously[1]. Thus, the bilinearity condition is dropped for the official definition and only sesquilinearity is required. The standard scalar product is defined as the sum of the products of the vector components with the associated complex conjugated vector components of the other vector. Let $\mathbf{x}_1$,$\mathbf{x}_2 \in \mathbb{C}^n$, then the scalar product is given by [1]: $\langle \mathbf{x}_1, \mathbf{x}_2\rangle := \sum_{k=1}^{n} x_{1k}\overline{x_{2k}}$. In contrast, we use a modified scalar product (marked by a "*") where, analogously to the real vector definition, the products of the vector components are summated: $\langle {}^*\mathbf{x}_1, \mathbf{x}_2\rangle := \sum_{k=1}^{n} x_{1k}x_{2k}$. This product (not a scalar product in strict mathematical sense) is a bilinear form but no longer positive definite. For real

[1]This can be verified by a simple calculation: Consider some vector $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{x} \in \mathbb{C}^n$. Since $\langle . \rangle$ is positive definite: $\langle \mathbf{x}, \mathbf{x}\rangle > 0$, by bilinearity: $\langle \sqrt{-i}\mathbf{x}, \sqrt{-i}\mathbf{x}\rangle = -i\langle \mathbf{x}, \mathbf{x}\rangle \not> 0$).

number vectors this modified scalar product is identical to the usual definition. With this modified scalar product we get

$$\begin{aligned}&\langle {}^*\Phi_c(\mathbf{x}_1), \Phi_c(\mathbf{x}_2)\rangle \\ =& -ax_{11}^2 - ax_{21}^2 + 2ax_{11}x_{21} - \cdots - ax_{1n}^2 - \\ & ax_{2n}^2 + 2ax_{1n}x_{2n} + c = -a||\mathbf{x}_1 - \mathbf{x}_2||^2 + c\end{aligned} \tag{13}$$

which is just the result of the NDK. The optimization can be done with the dual form. Thus, no complex number optimization is necessary. For the decision on the class to which a data vector should be assigned we switch to the primal form. The vector $\mathbf{w} \in \mathbb{C}^{4n+1}$ is calculated by: $\mathbf{w} := \sum_{j=1}^{m} \alpha_j y_j \Phi_c(\mathbf{x}_j)$ for all SVs $\mathbf{x}_j \in \mathbb{R}^n$. The decision function is then given by: $dc(\mathbf{x}) := \text{sgn}(\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle + b)$. Note that the modified scalar product $\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle$ must be a real number. This is stated in the following proposition.

Proposition 2 Let $\mathbf{w} = \sum_{j=1}^{m} \alpha_j y_j \Phi_c(\mathbf{x}_j)$ with $\mathbf{x}_j \in \mathbb{R}^n$, $\alpha_j \in \mathbb{R}$, $y_j \in \{-1, 1\}$, $j = 1, \ldots, m$, Φ_c as defined in formula (12) and $\mathbf{x} \in \mathbb{R}^n$. Then $\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle$ is a real number.

Proof: $\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle$ is given by:

$$\begin{aligned}\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle &= \langle {}^*\sum_{j=1}^{m} \alpha_j y_j \Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle \\ &= \sum_{j=1}^{m} \langle {}^*\alpha_j y_j \Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle \quad (\langle {}^*.\rangle \text{ is bilinear}) \\ &= \sum_{j=1}^{m} \alpha_j y_j \langle {}^*\Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle \\ &= \sum_{j=1}^{m} \alpha_j y_j (-a||\mathbf{x}_j - \mathbf{x}||^2 + c) \quad (\text{see form. (13)})\end{aligned} \tag{14}$$

which is clearly a real number. ∎

The NDK is related to the polynomial kernel since it can also be represented by a polynomial. However, it has the advantage over the polynomial kernel that it is faster to compute, since the number of dimensions in the target space grows only linearly and not exponentially with the number of dimensions in the original space [16], [2]. It remains to show that the decision functions following the primal and dual form are also equivalent for the modified form of the scalar product. This is stated in the follow proposition:

Proposition 3 Let $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^m$, $y \in \{-1, 1\}^m$, $\mathbf{w} := \sum_{j=1}^{m} \alpha_j y_j \Phi_c(\mathbf{x}_j)$ and $\langle {}^*\Phi_c(\mathbf{z}_1), \Phi_c(\mathbf{z}_2)\rangle = K(\mathbf{z}_1, \mathbf{z_2})$ $\forall \mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^n$. Then $\text{sgn}(\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle + b) = \text{sgn}(\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b)$.

Proof:

$$\begin{aligned}&\text{sgn}(\langle {}^*\mathbf{w}, \Phi_c(\mathbf{x})\rangle + b) \\ =& \text{sgn}(\langle {}^*\sum_{j=1}^{m} \alpha_j y_j \Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle + b) \\ =& \text{sgn}(\sum_{j=1}^{m} \langle {}^*\alpha_j y_j \Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle + b) \quad (\langle {}^*.\rangle \text{ is bilinear}) \\ =& \text{sgn}(\sum_{j=1}^{m} \alpha_j y_j \langle {}^*\Phi_c(\mathbf{x}_j), \Phi_c(\mathbf{x})\rangle + b) \\ =& \text{sgn}(\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + b) \\ =& \text{sgn}(\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b) \quad (K \text{ is symmetric})\end{aligned} \tag{15}$$

(16)

∎

Normally, feature vectors for document classification represent the weighted occurrences of lemma or word forms [15]. Thus, such a vector contains a large number of zeros and is therefore usually considered sparse. In this case, the computational complexity of the scalar product can be reduced from $\mathcal{O}(n)$ (where n is the number of vector components) to some constant runtime, which is the average number of non-zero vector components. Let I_1 be the set of indices of non-zero entries of vector $\mathbf{x}_1$ ($I_1 = \{k \in \{1, \ldots, n\} : x_{1k} \neq 0\}$) and I_2 be analogously defined for vector $\mathbf{x}_2$. The scalar product of both vectors can then be computed by $\sum_{k \in (I_1 \cap I_2)} x_{1k} \cdot x_{2k}$. Let us now consider the case that both vectors are transformed to complex numbers before the scalar multiplication. In this case, the modified scalar product

$$\begin{aligned}&\langle {}^*\Phi_c(\mathbf{x}_1), \Phi_c(\mathbf{x}_2)\rangle = \langle (\phi(x_{11}), \ldots, \phi(x_{1n})), \\ &(\phi(x_{21}), \ldots, \phi(x_{2n}))\rangle + c\end{aligned} \tag{17}$$

is considered where $\phi(x_k)$ denotes the transformation of a single real vector component to a complex number vector and is defined as:

$$\phi : \mathbb{R} \to \mathbb{C}^4, \ \phi(x_k) := (\sqrt{a}(x_k^2 - 1), \sqrt{a}i, \sqrt{2a}x_k, \sqrt{a}ix_k^2) \tag{18}$$

Note that the partial modified scalar product $\langle {}^*\phi(x_{1k}), \phi(x_{2k})\rangle$ can be non-zero, if at least one of the two vector components x_{1k} and x_{2k} is non-zero, which is easy to see:

$$\begin{aligned}&\langle {}^*\phi(x_k), \phi(0)\rangle = \sqrt{a}(x_k^2 - 1) \cdot \sqrt{a}(-1) + (-1)a \\ &+\sqrt{2a}x_k \cdot 0 + \sqrt{a}ix_k^2 \cdot 0 = a - ax_k^2 - a = -ax_k^2\end{aligned} \tag{19}$$

Only if both vector components are zero, one can be sure that the result is also zero:

$$\langle {}^*\phi(0), \phi(0)\rangle = (-\sqrt{a})(-\sqrt{a}) + ai \cdot i + 0 + 0 = a - a = 0 \tag{20}$$

Thus, the sparse data handling of two transformed complex vectors has to be modified in such a way that vector components associated with the union and not the intersection of non-zero indices are considered for multiplication:

$$\langle {}^*\mathbf{x}_1, \mathbf{x}_2\rangle = \sum_{k \in (I_1 \cup I_2)} \langle {}^*\phi(x_{1k}), \phi(x_{2k})\rangle + c \tag{21}$$

A further advantage of the NDK is that the Mahalanobis distance [8] can easily be integrated, which is shown in the remainder of this section. Each symmetric matrix $\mathbf{A}$ can be represented by the product $\mathbf{V}^{-1}\mathbf{D}\mathbf{V}$ where $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{A}$ on its diagonals. The square root of a matrix $\mathbf{A}$ is then defined as $\sqrt{\mathbf{A}} := \mathbf{V}^{-1}\mathbf{D}^{0.5}\mathbf{V}$ where $\mathbf{D}^{0.5}$ is the matrix with the square root of the eigenvalues of $\mathbf{A}$ on its diagonal. It is obvious

that $\sqrt{\mathbf{A}} \cdot \sqrt{\mathbf{A}} = \mathbf{A}$.

Proposition 4: Let $\mathbf{x}$, $\mathbf{z} \in \mathbb{R}^n, a := 1, c := 0$, then $\langle {}^*\Phi_c(\sqrt{\mathbf{Cov}^{-1}}\mathbf{x}), \Phi_c(\sqrt{\mathbf{Cov}^{-1}}\mathbf{z})\rangle = -MH(\mathbf{x},\mathbf{z})^2$ where $MH(\mathbf{x},\mathbf{z})$ denotes the Mahalanobis distance $\sqrt{(\mathbf{x}-\mathbf{z})^T\mathbf{Cov}^{-1}(\mathbf{x}-\mathbf{z})}$ and $\mathbf{Cov}$ denotes the covariance matrix between the feature values of the entire training data set.

Proof: We have that ($\mathbf{with\ C} := \sqrt{\mathbf{Cov}^{-1}}$):

$$\begin{aligned}
&\langle {}^*\Phi_c(\mathbf{Cx}), \Phi_c(\mathbf{Cz})\rangle \\
&= -\,||\mathbf{Cx}-\mathbf{Cz}||^2 \qquad (\text{see formula (13)}) \\
&= -\,(\mathbf{Cx}-\mathbf{Cz})^\top(\mathbf{Cx}-\mathbf{Cz}) \\
&= -\,(\mathbf{C}(\mathbf{x}-\mathbf{z}))^\top(\mathbf{C}(\mathbf{x}-\mathbf{z})) \\
&= -\,(\mathbf{x}-\mathbf{z})^\top\sqrt{\mathbf{Cov}^{-1}}^\top\sqrt{\mathbf{Cov}^{-1}}(\mathbf{x}-\mathbf{z}) \\
&= -\,(\mathbf{x}-\mathbf{z})^\top\mathbf{Cov}^{-1}(\mathbf{x}-\mathbf{z})
\end{aligned} \tag{22}$$

(since the covariance matrix is symmetric and the inverse and square root of a symmetric matrix is also symmetric)

■

This proposition shows that the NDK is just the negative square of the Mahalanobis distance for some vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ if $\hat{\mathbf{x}}$ is set to $\sqrt{\mathbf{Cov}^{-1}}\mathbf{x}$ (analogously for $\hat{\mathbf{z}}$), a is set to 1 and c is set to 0. Furthermore, this proposition shows that we can easily extend our primal form to integrate the Mahalanobis distance. For that, we define a function $\tau : \mathbb{R}^n \to \mathbb{R}^n$ as follows: $\tau(\mathbf{x}) = \sqrt{\mathbf{Cov}^{-1}}\mathbf{x}$. With Φ_m:=$\Phi_c \circ \tau$ we have:

$$\langle {}^*\Phi_m(\mathbf{x}), \Phi_m(\mathbf{z})\rangle = (MH(\mathbf{x},\mathbf{z}))^2 \tag{23}$$

which shows that we have indeed derived a primal form.

We use the NDK for text classification where a text is automatically labeled regarding its main topics (i.e., categories), employing the DDC as one of the leading classification schemes in digital libraries [11]. Documents are classified regarding the 10 top-level categories of the DDC. To this end, we use training sets of documents for which the DDC categories have already been assigned. Lexical features are extracted from this training set and made input to an SVM library (i.e., $libsvm$ [3]) in order to assign one or more DDC categories to previously unseen texts. Features of instance documents are computed by means of the geometric mean of the tfidf values and the GSS coefficients of their lexical constituents [4], [13].

EVALUATION

The evaluation and training was done using 4 000 German documents, containing in total 114 887 606 words and 9 643 022 sentences requiring a storage space of 823.51 MB. There are 400 documents of each top-level DDC category in the corpus. 2 000 of the texts were used for training, 2 000 for evaluation. The corpus consists of texts annotated according to the OAI (*Open Archive Initiative*) and is presented in [7]. We tested the correctness of category assignment for the 10 top-level DDC categories. Each document is mapped at least to one DDC category. Multiple assignments are possible. Precision, recall and F-scores were computed for each DDC category using the NDK, the square (polynomial kernel of degree 2), the cubic (polynomial kernel of degree 3), the RBF kernel and the linear kernel (see Tables I and II). The free parameters of the square, the cubic and the RBF kernel are determined by means of a grid search on an independent data set. On the same held-out dataset, we adjusted the SVM-threshold parameter b to optimize the F-scores. The time (on an Intel Core i7) required to obtain the classifications was determined (see Table III for the real / complex NDK, the RBF, square, and linear kernel). The time needed for building the model was not measured because of being required only once and therefore being irrelevant for online text classification.

The F-score for the NDK is higher than the F-scores of all other kernels and faster to compute except for the linear kernel – obviously, the classification using the primal form of the linear kernel is faster than the one using the NDK. Furthermore, the complex decomposition of the NDK leads to a considerable acceleration of the classification process compared to its dual form and should therefore be preferred.

We conducted a second experiment where we compared the linear kernel with the NDK now using text snippets (i.e., abstracts instead of full texts) taken once more from our OAI corpus. This application scenario is more realistic for digital libraries that mainly harvest documents by means of their meta data and, therefore, should also be addressed by text classifiers. The kernels were trained on a set of 19 000 abstracts[2]. This time, the categories are not treated independently of each other. Instead, the classifier selected all categories with signed distance values from the SVM hyperplane that are greater or equal to zero. If all distance values are negative, the category with the largest (smallest absolute) distance was chosen. In this experiment, the linear kernel performed better than the NDK. Using three samples of 500 texts, the F-scores of the linear kernel are (macro-averaged over all main 10 DDC categories): 0.753, 0.735, 0.743, and of the NDK: 0.730, 0.731, 0.737. This result indicates that the heuristic of preferring the categories of highest signed distance to the hyperplane is not optimal for the NDK. We compared our classifier with two other systems, the DDC classifier of [18] and of [6]. [18] reaches an F-score of 0.502, 0.457 and 0.450 on these three samples. The F-scores of the DDC-classifier of [6] are 0.615, 0.604, and 0.574. Obviously, our classifier outperforms these competitors.

Finally, we evaluated the NDK on the Reuters 21578 corpus[3] (Lewis split). This test set is very challenging since 35 of all 93 categories occurring in this split have 10 or less training examples. Furthermore, several texts are intentionally assigned to none of the Reuters categories. We created an artificial category for all texts that are not assigned in this sense. The histogram of category assignments is displayed in Figure 1. The F-scores for the Reuters corpus are given in Table IV. We modified the training data by filling up instances for every category by randomly selecting positive instances such that the ratio of positive examples to all instances are the same for all categories. Hence, in most category samples some training examples were used more than once. This approach prevents from preferring categories due to their multitude of training examples. The F-score of the NDK is lower than the

[2]Note that in this evaluation we employed the tfidf score for weighting only, since the use of the GSS coefficient didn't lead to any improvement in F-score.

[3]URL: http://www.daviddlewis.com/resources/testcollections/reuters21578

TABLE I. PRECISION / RECALL OF DIFFERENT KERNELS.

	NDK		RBF		Linear		Square		Cubic	
Cat.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Recall
0	0.845	0.777	0.867	0.731	0.747	0.853	0.831	0.772	0.789	0.817
1	0.768	0.740	0.775	0.682	0.726	0.745	0.722	0.771	0.712	0.734
2	0.909	0.833	0.892	0.857	0.905	0.847	0.873	0.882	0.935	0.635
3	0.764	0.497	0.679	0.571	0.545	0.640	0.716	0.508	0.708	0.487
4	0.852	0.529	0.870	0.388	0.863	0.490	0.441	0.146	0.000	0.029
5	0.700	0.783	0.794	0.626	0.825	0.675	0.744	0.660	0.697	0.793
6	0.682	0.685	0.718	0.625	0.699	0.685	0.670	0.700	0.744	0.595
7	0.680	0.741	0.675	0.652	0.602	0.751	0.722	0.697	0.843	0.532
8	0.657	0.720	0.586	0.785	0.626	0.774	0.670	0.677	0.752	0.457
9	0.701	0.752	0.680	0.670	0.668	0.723	0.775	0.636	0.831	0.335
All	0.756	0.706	0.754	0.659	0.721	**0.718**	0.716	0.645	**0.801**	0.542

TABLE II. F-SCORES OF DIFFERENT KERNELS EVALUATED BY MEANS OF THE OAI CORPUS OF [7].

Cat.	NDK	Square	Cubic	RBF	Linear
0	0.810	0.800	0.803	0.793	0.796
1	0.753	0.746	0.723	0.726	0.735
2	0.869	0.877	0.757	0.874	0.875
3	0.603	0.594	0.577	0.621	0.589
4	0.653	0.219	0.057	0.537	0.625
5	0.740	0.700	0.742	0.700	0.743
6	0.683	0.685	0.661	0.668	0.692
7	0.710	0.709	0.652	0.663	0.668
8	0.687	0.674	0.569	0.671	0.692
9	0.726	0.699	0.478	0.675	0.695
All	**0.723**	0.670	0.602	0.693	0.711

TABLE III. TIME (IN MILLISECONDS) REQUIRED FOR THE COMPUTATION OF THE KERNELS ON THE OAI CORPUS.

C.	NDK prim.	NDK dual	Squ. dual	Cubic dual	RBF dual	Lin. dual
0	**6 426**	46 200	50 256	50 979	47 246	44 322
1	**9 339**	98 289	115 992	132 844	122 751	93 550
2	**5 822**	48 359	57 603	67 200	65 508	46 786
3	**23 774**	134 636	165 604	177 211	159 897	111 516
4	**10 103**	118 322	77 634	93 613	111 153	103 548
5	**32 501**	104 865	72 772	135 820	138 107	96 637
6	**24 375**	105 614	116 892	120 919	126 498	95 423
7	**19 206**	113 971	122 128	143 562	120 371	100 640
8	**11 236**	99 288	106 215	118 378	105 942	84 381
9	**20 822**	125 358	139 004	168 837	138 395	111 245
all	**16 361**	99 490	102 410	120 936	113 587	88 805

TABLE IV. MEAN F-SCORE, PRECISION, AND RECALL (MACRO-AVERAGING) OF DIFFERENT KERNELS EVALUATED BY MEANS OF THE REUTERS CORPUS.

Kernel	F-score	Precision	Recall
NDK	0.394	0.414	0.419
Square	0.324	0.392	0.304
Cubic	0.348	0.319	**0.567**
RBF	0.403	0.420	0.441
Linear	**0.408**	**0.428**	0.436

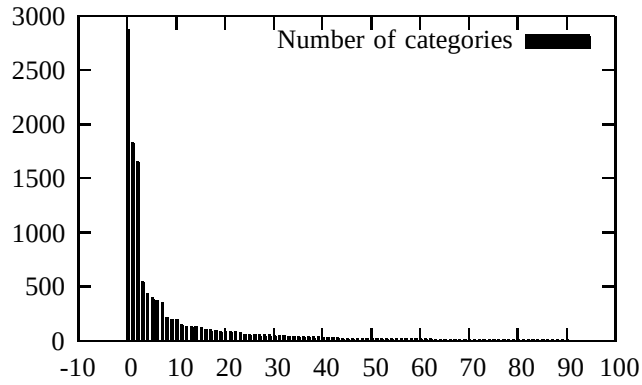

Fig. 1. Histogram of the distribution of categories for the Reuters corpus.

highest macro-averaging F-score obtained by the linear kernel, but outperforms the square and the cubic kernel. Again, the parameters of the NDK, the square, the cubic and the RBF kernels were determined by means of a grid search.

CONCLUSION AND FUTURE WORK

We derived a primal form of the NDK by means of complex numbers. In this way, we obtained a much simpler representation compared to the modified dual form. We showed that the primal form (and in principle also the modified dual form) can speed up text classification considerably. The reason is that it does not require to compare input vectors with all support vectors. Our evaluation showed that the F-scores of the NDK are competitive regarding all other kernels tested here while the NDK consumes less time than the polynomial and the RBF kernel. We have also shown that the NDK performs better than the linear kernel when using full texts rather than text snippets. Whether this is due to problems of feature selection/expansion or a general characteristic of this kernel (in the sense of being negatively affected by ultra-sparse features spaces), will be examined in future work. Additionally, we plan to examine the PK with exponents larger than two, to investigate under which prerequisites the PK performs well and to evaluate the extension of the NDK that includes the Mahalanobis distance.

ACKNOWLEDGEMENT

We thank Vincent Esche and Tom Kollmar for valuable comments and suggestions for improving our paper.

REFERENCES

[1] Albrecht Beutelsbacher. *Lineare Algebra*. Vieweg, Braunschweig, Germany, 2010.

[2] Shri D. Boolchandani and Vineet Sahula. Exploring efficient kernel functions for support vector machine based feasability models for analog circuits. *International Journal of Design Analysis and Tools for Circuits and Systems*, 1(1):1–8, 2011.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

[4] Thorsten Joachims. *The Maximum Margin Approach to Learning Text Classifiers: Methods, Theory, and Algorithms*. PhD thesis, Universität Dortmund, Informatik, LS VIII, 2000.

[5] Thorsten Joachims. *Learning to classify text using support vector machines*. Kluwer, Boston, 2002.

[6] Mathias Lösch. Talk: Automatische Sacherschließung elektronischer Dokumente nach DDC, 2011.

[7] Mathias Lösch, Ulli Waltinger, Wolfram Horstmann, and Alexander Mehler. Building a DDC-annotated corpus from OAI metadata. *Journal of Digital Information*, 12(2), 2011.

[8] Prasanta Chandra Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*, pages 49–55, 1936.

[9] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[10] Alexander Mehler and Ulli Waltinger. Enhancing document modeling by means of open topic models: Crossing the frontier of classification schemes in digital libraries by example of the DDC. *Library Hi Tech*, 27(4):520–539, 2009.

[11] OCLC. Dewey decimal classification summaries. a brief introduction to the dewey decimal classification, 2012. URL: http://www.oclc.org/dewey/resources/summaries/default.htm, last access: 8/17/2012.

[12] Hichem Sahbi and Francois Fleuret. Scale-invariance of support vector machines based on the triangular kernel. Technical Report 4601, Institut National de Recherche en Informatique et an Automatique, 2002.

[13] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 25(5):513–523, 1998.

[14] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, Massachusetts, 2002.

[15] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[16] César Souza. Kernel functions for machine learning applications, 2010. url:http://crsouza.blogspot.de/2010/03/kernel-functions-for-machine-learning.html.

[17] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, New York, 1998.

[18] Ulli Waltinger, Alexander Mehler, Mathias Lösch, and Wolfram Horstmann. Hierarchical classification of oai metadata using the DDC taxonomy. In *Advanced Language Technologies for Digital Libraries*, Lecture Notes in Computer Science, pages 29–40. Springer, Heidelberg, Germany, 2011.

[19] Shih-Hung Wu. Support vector machine tutorial, 2012. URL=www.csie.cyut.edu.tw/~shwu/PR_slide/SVM.pdf.