Pattern Recognition 71 (2017) 187–195

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Indefinite Core Vector Machine

Frank-Michael Schleif[a,b,*], Peter Tino[a]

[a] School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
[b] University of Applied Sciences Wuerzburg Schweinfurt, Sanderheinrichsleitenweg 20, 97074 Wuerzburg, Germany

ARTICLE INFO

Article history:
Received 27 December 2016
Revised 6 April 2017
Accepted 1 June 2017
Available online 3 June 2017

Keywords:
Indefinite learning
Krĕin space
Classification
Core Vector Machine
Nyström
Sparse
Linear complexity

ABSTRACT

The recently proposed Krĕin space Support Vector Machine (KSVM) is an efficient classifier for indefinite learning problems, but with quadratic to cubic complexity and a non-sparse decision function. In this paper a Krĕin space Core Vector Machine (iCVM) solver is derived. A sparse model with linear runtime complexity can be obtained under a low rank assumption. The obtained iCVM models can be applied to indefinite kernels without additional preprocessing. Using iCVM one can solve CVM with usually troublesome kernels having large negative eigenvalues or large numbers of negative eigenvalues. Experiments show that our algorithm is similar efficient as the Krĕin space Support Vector Machine but with substantially lower costs, such that also large scale problems can be processed.

1. Introduction

Learning of classification models for indefinite kernels received substantial interest with the advent of domain specific similarity measures. Indefinite kernels are a severe problem for most kernel based learning algorithms because classical mathematical assumptions such as positive definiteness, used in the underlying optimization frameworks are violated. As a consequence e.g. the classical Support Vector Machine (SVM) [1] has no longer a convex solution - in fact, most standard solvers will not even converge for this problem [2]. Researchers in the field of e.g. psychology [3], vision [4–6] and machine learning [7,8] have criticized the typical restriction to metric similarity measures. In fact in [8] for multiple examples from real problems it is shown that many real life problems are better addressed by e.g. kernel functions which are not restricted to be based on a metric. Non-metric measures (leading to kernels which are not positive semi-definite (non-psd)) are common in many disciplines. The use of divergence measures [9–11] is very popular for spectral data analysis in chemistry, geo- and medical sciences [12,13], and are in general not metric. Also the popular Dynamic Time Warping (DTW) [14] algorithm provides a non-metric alignment score which is often used as a proximity measure between two one-dimensional functions of different length. In image processing and shape retrieval indefinite proximities are often obtained by means of the inner distance [15] - another non-metric measure. Further examples can be found in physics, where problems of the special relativity theory naturally lead to indefinite spaces. Further prominent examples for genuine non-metric proximity measures can be found in the field of bioinformatics where classical sequence alignment algorithms (e.g. smith-waterman score [16]) produce non-metric proximity values. Multiple authors argue that the non-metric part of the data contains valuable information and should not be removed [6,7].

Furthermore, it has been shown [2,7,17] that work-arounds such as eigenspectrum modifications are often inappropriate or undesirable, due to a loss of information and problems with the out-of sample extension.

Due to its strong theoretical foundations, Support Vector Machine (SVM) has been extended for indefinite kernels in a number of ways [18–20]. Initial work focused on preprocessing the kernel matrix through heuristics to address the indefiniteness [21]. A recent survey on indefinite learning is given in [17]. In [2] a stabilization approach was proposed to calculate a valid SVM model in the Krĕin space which can be directly applied on indefinite kernel matrices. This approach has shown great promise in a number of learning problems but has intrinsically quadratic to cubic complexity and provides a dense decision model. This paper extends the work of [2] by deriving an equivalent optimization problem but within the Core Vector Machine (CVM) framework [22]. To ensure linear runtime complexity we combine the proposed

* Corresponding author at: School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK.
E-mail addresses: schleif@informatik.uni-leipzig.de, schleify@cs.bham.ac.uk, fmschleif@googlemail.com, fschleif@techfak.uni-bielefeld.de (F.-M. Schleif), pxt@cs.bham.ac.uk (P. Tino).

http://dx.doi.org/10.1016/j.patcog.2017.06.003
0031-3203/

indefinite CVM with a low rank kernel approximation using the Nyström approach [23]. The latter one will also serve as a key element to sparsify the final solution such that an easy out of sample extension becomes possible. We empirically demonstrate the effectiveness of the proposed approach in comparison to the KSVM.

1.1. Indefinite kernels and existing approaches

Domain specific proximity measures, such as alignment scores in bioinformatics [24], the edit-distance for structural pattern recognition [25], shape retrieval measures (e.g. the inner distance [15]) and many other ones, generate non-metric or indefinite similarities or dissimilarities. Classical learning algorithms such as kernel machines assume metric properties in the underlying data space and may not be applicable for this type of data.

Only few machine learning methods have been proposed for non-metric proximity data, e.g. the indefinite kernel fisher discriminant (iKFD) [26,27] or the probabilistic classification vector machine (PCVM) [28]. The iKFD is a classical fisher discriminant approach, maximizing the between class variance of the classes, but formulated in the Krĕin space, by using an equivalence relation to the classical kernel Fisher Discriminant Analysis.[1] In its original formulation, iKFD provides models which are naturally non-sparse and has cubic runtime complexity. The PCVM, on the other hand, constitutes a probabilistic model, operating with basis functions in the input space without the need for the existence of feature space (through Mercer kernel). While the iKFD is a batch optimization algorithm the PCVM is formulated by a gradient descent strategy with potentially slow convergence for a number of problems. The PCVM algorithms has cubic complexity in the first iterations with a substantial speed-up during further iterations due to an inherent sparsification strategy.

Recently the Krĕin space Support Vector Machine (KSVM) was proposed in [2] leading to an SVM equivalent formulation, but fully formalized in the Krĕin space by replacing the SVM minimization problem with a stabilization problem. As shown in [2] it turns out that solving the stabilization problem (detailed in [2],sec 2) can be achieved by flipping the negative eigenvalues of the kernel spectrum. It is shown in [2] that this strategy has a theoretical foundation and by solving the stabilization problem one can obtain the solution in the original Krĕin space. This allows us to classify any new point without having to transform it.iKFD and PCVM have been found to be very effective but unlike KSVM, they are not based on the sound theoretical framework of the SVM structural risk minimization principle (SRM) [1]. Furthermore, there are a number of other advantages of KSVM as outlined in [2]. Hence, it is very attractive to obtain a low cost SVM formulation in the Krĕin space, which is the focus of this paper.

1.2. Contributions

We consider the problem of training a Core Vector Machine with an indefinite kernel. The present paper is based on [2] in which the stabilization idea is proposed and on effective Nyström approximation concepts given in [34], both applicable to indefinite kernels. To ensure linear runtime complexity in contrast to at least quadratic costs of the KSVM we derive an indefinite Core Vector Machine using a low rank kernel approximation which solves the original indefinite SVM problem at low costs. We also suggest a sparsification procedure to simplify the out of sample extension. The Nystöm approximation is not necessary to obtain an indefinite Core Vector Machine, but to keep linear runtime and memory complexity which is lost otherwise.

[1] We do not detail the approach here because the paper will focus on an extension of KSVM.

2. Krĕin space SVM

The Krĕin Space SVM (KSVM) [2], replaced the classical SVM minimization problem by a stabilization problem in the Krĕin space. The respective equivalence between the stabilization problem and a standard convex optimization problem was shown in [2]. Let $x_i \in X, i \in \{1,\ldots,N\}$ be training points in the input space X , with labels $y_i \in \{-1,1\}$, representing the class of each point. The input space X is often considered to be $\mathbb{R}^d$, but can be any suitable space due to the kernel trick. For a given positive C, SVM is the minimum of the following regularized empirical risk functional

$$J_C(f,b) = \min_{f\in\mathcal{H},b\in\mathbb{R}} \frac{1}{2}\|f\|^2_{\mathcal{H}} + CH(f,b) \tag{1}$$

$$H(f,b) = \sum_{i=1}^{N} \max(0, 1 - y_i(f(x_i)+b))$$

Using the solution of Eq. (1) as $(f_C^*, b_C^*) := \arg\min J_C(f,b)$ one can introduce $\tau = H(f_C^*, b_C^*)$ and the respective convex quadratic program (QP)

$$\min_{f\in\mathcal{H},b\in\mathbb{R}} \frac{1}{2}\|f\|^2_{\mathcal{H}} \quad s.t. \sum_{i=1}^{N} \max(0, 1 - y_i(f(x_i)+b)) \le \tau \tag{2}$$

where we detail the notation in the following. This QP can be also seen as the problem of retrieving the orthogonal projection of the null function in a Hilbert space $\mathcal{H}$ onto the convex feasible set. The view as a projection will help to link the original SVM formulation in the Hilbert space to a KSVM formulation in the Krein space. First we need to repeat a few definitions, widely following [2]. A Krĕin space is an *indefinite* inner product space endowed with a Hilbertian topology.

Definition 1 (Inner products and inner product space)**.** Let $\mathcal{K}$ be a real vector space. An inner product space with an indefinite inner product $\langle\cdot,\cdot\rangle_{\mathcal{K}}$ on $\mathcal{K}$ is a bi-linear form where all $f,g,h \in \mathcal{K}$ and $\alpha \in \mathbb{R}$ obey the following conditions:

Symmetry: $\langle f,g\rangle_{\mathcal{K}} = \langle g,f\rangle_{\mathcal{K}}$,linearity: $\langle \alpha f + g, h\rangle_{\mathcal{K}} = \alpha\langle f,h\rangle_{\mathcal{K}} + \langle g,h\rangle_{\mathcal{K}}$and $\langle f,g\rangle_{\mathcal{K}} = 0\ \forall g \in \mathcal{K}$ implies $f = 0$.

An inner product is positive definite if $\forall f \in \mathcal{K},\ \langle f,f\rangle_{\mathcal{K}} \ge 0$, negative definite if $\forall f \in \mathcal{K},\ \langle f,f\rangle_{\mathcal{K}} \le 0$, otherwise it is indefinite. A vector space $\mathcal{K}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{K}}$ is called inner product space.

Definition 2 (Krĕin space and pseudo Euclidean space)**.** An inner product space $(\mathcal{K},\langle\cdot,\cdot\rangle_{\mathcal{K}})$ is a Krĕin space if there exist two Hilbert spaces $\mathcal{H}_+$ and $\mathcal{H}_-$ spanning $\mathcal{K}$ such that $\forall f \in \mathcal{K},\ f = f_+ + f_-$ with $f_+ \in \mathcal{H}_+,\ f_- \in \mathcal{H}_-$ and $\forall f,g \in \mathcal{K},\ \langle f,g\rangle_{\mathcal{K}} = \langle f_+,g_+\rangle_{\mathcal{H}_+} - \langle f_-,g_-\rangle_{\mathcal{H}_-}$. A finite-dimensional Krĕin-space is a so called pseudo Euclidean space (pE).

If $\mathcal{H}_+$ and $\mathcal{H}_-$ are reproducing kernel hilbert spaces (RKHS), $\mathcal{K}$ is a reproducing kernel Krĕin space (RKKS). For details on RKHS and RKKS see e.g.[35]. In this case the uniqueness of the functional decomposition (the nature of the RKHSs $\mathcal{H}_+$ and $\mathcal{H}_-$) is not guaranteed. In [36] the reproducing property is shown for a RKKS $\mathcal{K}$. There is a unique symmetric kernel $k(x,x)$ with $k(x,\cdot) \in \mathcal{K}$ such that the reproducing property holds (for all $f \in \mathcal{K}, f(x) = \langle f, k(x,\cdot)\rangle_{\mathcal{K}}$) and $k = k_+ - k_-$ where k_+ and k_- are the reproducing kernels of the RKHSs $\mathcal{H}_+$ and $\mathcal{H}_-$.

As shown in [36] for any symmetric non-positive kernel k that can be decomposed as the difference of two positive kernels k_+ and k_-, a RKKS can be associated to it. In [2] it was shown how the classical SVM problem can be reformulated by means of a stabilization problem. This is necessary because a classical norm as used in Eq. (2) does not exist in the RKKS but instead the norm is reinterpreted as a projection which still holds in RKKS and is

used as a regularization technique [2]. This allows to define SVM in RKKS (viewed as Hilbert space) as the orthogonal projection of the null element onto the set [2]:

$$S = \{f \in \mathcal{K}, b \in \mathbb{R} | H(f,b) \leq \tau\} \text{ and } 0 \in \partial_b H(f,b)$$

where ∂_b denotes the sub differential with respect to b. The set S leads to a unique solution for SVM in a Krĕin space [2]. As detailed in [2] one finally obtains a stabilization problem which allows one to formulate an SVM in a Krĕin space.

$$\text{stab}_{f\in\mathcal{K}, b\in\mathbb{R}} \frac{1}{2}\langle f, f\rangle_{\mathcal{K}} \quad \text{s.t.} \sum_{i=1}^{l} \max(0, 1 - y_i(f(x_i) + b)) \leq \tau \qquad (3)$$

where stab means stabilize as detailed in the following: In a classical SVM in RKHS the solution is regularized by minimizing the norm of the function f. In Krĕin spaces however minimizing such a norm is meaningless since the dot-product contains both the positive and negative components. Thats why the regularization in the original SVM through minimizing the norm f has to be transformed in the case of Krĕin spaces into a min-max formulation, where we jointly minimize the positive part and maximize the negative part of the norm. The authors of [36] termed this operation the stabilization projection, or stabilization. Further mathematical details can also be found in [37,38]. An example illustrating the relations between minimum, maximum and the projection/stabilization problem in the Krĕin space is illustrated in [2].

In [2] it is further shown that the stabilization problem Eq. (3) can be written as a minimization problem using a semi-definite kernel matrix. By defining a projection operator with transition matrices it is also shown how the dual RKKS problem for the SVM can be related to the dual in the RKHS. We refer the interested reader to [2]. One - finally - ends up with a flipping operator applied to the eigenvalues of the indefinite kernel matrix[2] K as well as to the α parameters obtained from the stabilization problem in the Krĕin space, which can be solved using classical optimization tools on the flipped kernel matrix. This permits to apply the obtained model from the Krĕin space directly on the non-positive input kernel without any further modifications. The algorithm is shown in Algorithm 1. There are four major steps: (1) an eigen-

Algorithm 1 Krĕin Space SVM (KSVM) - adapted from [2].

Krĕin SVM:
$[U, D]$ = EigenDecomposition(K)
$\hat{K} = USDU^\top$ with $S = sign(D)$ $\hat{K}$ is not generated as a full matrix for iCVM - see text
$[\alpha, b]$ = SVMSolver($\hat{K}$, Y, C)
$\tilde{\alpha} = USU^\top\alpha$
return $\tilde{\alpha}, b$;

decomposition of the full kernel matrix, with cubic costs (which can be potentially restricted to a few dominating eigenvalues - referred to as KSVM-L); (2) a flipping operation; (3) the solution of an SVM solver on the modified input matrix; (4) the application of the projection operator obtained from the eigen-decomposition on the α vector of the SVM model.

U in Algorithm. 1 contains the eigenvectors, D is a diagonal matrix of the eigenvalues and S is a matrix containing only $\{1, -1\}$ on the diagonal as obtained from the respective function sign.

As pointed out in [2], this solver produces an exact solution for the stabilization problem. The main weakness of this Algorithm is, that it requires the user to pre-compute the whole kernel matrix and to decompose it into eigenvectors/eigenvalues. Further today's SVM solvers have a theoretical, worst case complexity of $\approx \mathcal{O}(N^2)$. The other point to mention is that the final solution $\tilde{\alpha}$ is not sparse. We will address these points in the following.

3. Core Vector Machine

The Core Vector Machine (CVM) was initially proposed in [22] and it was shown that the SVM can be formulated as a minimum enclosing ball (MEB) problem leading to the CVM algorithm. This can be also done for arbitrary positive semi-definite (psd) input kernels as shown in [39]. The CVM is a very efficient algorithm providing accurate classification models for large scale data at very low costs [39], having a theoretical, worst case complexity of $\approx \mathcal{O}(N)$.

In [40] the CVM approach was criticized to be less stable then expected under some conditions, but this was partially caused by an error in the implementation and also due to the use of additional optimization tricks[3]. To use the given (potentially approximated) non-psd input kernel for CVM we have to modify the kernel matrix. Let K be the (approximated) indefinite kernel matrix. The kernel is adapted to a two class classification problem as done in [22][4]:

$$K' = Y \otimes K + Y + \frac{\delta}{C} \qquad (4)$$

where Y is the label matrix with entries $Y_{i,j} = y_i y_j$, $\otimes$ is the element-wise multiplication and $\frac{\delta}{C}$ is an all zero indicator matrix with non-vanishing entries $\frac{1}{C}$ only on the diagonal. C is the user defined SVM penalty parameter. In the second step we modify the kernel K' with respect to the approach suggested in KSVM to link the original CVM minimization problem to the stabilization problem (see [2]) as shown in Algorithm 1. This leads to a psd kernel $\hat{K}$ for a two-class classification problem as necessary for the CVM solver. Once more it should be noted that the eigenvalue correction used in Algorithm 1 is a natural consequence of the underlying stabilization procedure as detailed in [2] and not a heuristic choice. The final model can be applied on an *unmodified* indefinite kernel.

If $\hat{K}$ has constant values on the diagonal, it was shown in [39] that this directly leads to a MEB optimization problem. If $\hat{K}_{i,i}$ is non-constant, the generalized CVM can be used [39]. To avoid high computational costs the kernel matrix K' in (4) or the respective modification $\hat{K}$ (by using the KSVM methodology) can be approximated by the Nyström approximation as detailed in Section 4, including an eigen-decomposition with linear costs. Subsequently we give a few elementary details for the CVM algorithm to link it with our problem.

It has been shown e.g. in [41] that the minimum enclosing ball (MEB) can be approximated with quality ϵ in (worst case) linear time using an algorithm which requires only a constant subset of the training set R_j (a region), refereed to as the core set. Given fixed quality ϵ, the following algorithm converges in $\mathcal{O}(1/\epsilon^2)$ steps:

Here we assume that $\hat{K}_{i,j} = \langle \Phi(x_i), \Phi(x_k)\rangle$, such that the involved distance calculation can be expressed solely through inner products. Accordingly no explicit feature space is needed. In each step, the MEB problem is solved for a small subset of constant size only. This is possible by referring to the dual problem which has

[2] Obtained by evaluating $k(x, y)$ for training points x, y.

[3] Our code is matlab based with well tested numerical routines and we make use of a classical quadratic problem solver, instead of the sequential minimal optimization (SMO) approach. The SMO approach contains some selection and stopping heuristics which can be problematic sometimes.

[4] In case of multiclass problems we rely on a one vs rest approach using the same procedure.

the form

$\min_{\alpha_i \geq 0} \quad \sum_{ij} \alpha_i \alpha_j k_{ij} - \sum_i \alpha_i k_{ii}^2$

where $\quad \sum_i \alpha_i = 1$

with data points occurring only as dot products, i.e. kernelization is possible. The same holds for all distance computations of the approximate MEB problem. Note that the dual MEB problem provides a solution in terms of the dual variables α_i. By construction, each class is represented by at least two core points. The bias term of the SVM / CVM solution is obtained by $b = \sum \alpha_i y_i$. The set S obtained in Algorithm 2 can be potentially shrinked during the iteration by removing those x_i which have an α_i close to zero (e.g. $1e^{-10}$)[5]

Algorithm 2 MEB - Solver.

MEB:
$S := \{x_i, x_k\}$ for a pair of largest distance $\|\Phi(x_i) - \Phi(x_k)\|^2$ in R_j and x_i chosen randomly
repeat
  solve MEB$(S) \rightarrow \tilde{w}_j, R$
  if exists $x_l \in R_j$ where $\|\Phi(x_l) - \tilde{w}_j\|^2 > R^2(1+\epsilon)^2$ **then**
    $S := S \cup \{x_l\}$
  end if
until all x_l are covered by the $R(1+\epsilon)$ ball in the feature space
return $\tilde{w}_j$, radius R

4. Linear time eigen-decomposition for low rank matrices

The strategy described above is still based on the calculation of an eigen-decomposition of the kernel matrix K or K', with cubic costs for the full eigen-decomposition. Assuming that the original input kernel has low rank, the Nyström approximation can be used, which can also be applied to indefinite kernels [34]. The Nyström approximation for kernel methods (details in [23]) gives:

$$\tilde{K} = K_{(N,m)} K_{(m,m)}^{-1} K_{(m,N)}. \tag{5}$$

Thereby m (columns/rows) of the original kernel matrix have been selected as so called landmarks. The matrix $K_{(N,\,m)}$ consists of the m columns of the original kernel matrix with indices taken from the selected landmarks. $K_{(m,m)}^{-1}$ denotes the Moore-Penrose pseudo-inverse of the respective landmark matrix $K_{(m,\,m)}$. Strategies for landmark selection have been widely analyzed in recent times with most promising results by using leverage scores [42] or by adding pseudo landmarks [43]. To simplify the analysis we select the landmarks randomly and i.i.d. More complicated selection schemes, with potentially additional costs, will likely improve the results but are not in the main focus of this work. The approximation is exact, if $K_{(m,\,m)}$ has the same rank as K. Besides using the standard Nyström approximation to approximate a kernel matrix as in Eq. (5), a linear time eigenvalue correction for (potentially indefinite) low rank matrices was proposed in [34].

This low rank eigenvalue decomposition is used in the indefinite Core Vector Machine approach to approximate the respective kernel matrix. This is necessary to apply the flipping eigenspectrum correction with linear costs instead of cubic costs (or slightly less if the number of eigenvalues is restricted) in the KSVM-L approach.

For a matrix approximated by Eq. (5) it is possible to compute its exact eigenvalue decomposition in linear time[6]. Subsequently we review the concepts from [34], in particular an approach to obtain an eigendecomposition of a Nyström based indefinite matrix.

To compute the eigenvectors and eigenvalues of an *indefinite* matrix we first compute the squared form of the Nyström approximated kernel matrix. Let K be a similarity matrix, for which we can write its decomposition as

$$\tilde{K} = K_{(N,m)} K_{(m,m)}^{-1} K_{(m,N)} = K_{(N,m)} U \Lambda^{-1} U^\top K_{(N,m)}^\top \qquad = BB^\top,$$

where we defined $B = K_{(N,m)} U \Lambda^{-1/2}$ with U and Λ being the eigenvectors and eigenvalues of $K_{(m,\,m)}$, respectively. Further it follows for the *squared* $\tilde{K}$: $\tilde{K}^2 = BB^\top BB^\top = BVAV^\top B^\top$, where V and A are the eigenvectors and eigenvalues of $B^\top B$, respectively. Apparently the square operation does not change the eigenvectors of K but only the eigenvalues. The corresponding eigenequation can be written as $B^\top B v = av$. Multiplying with B from left we get: $\underbrace{BB^\top}_{\tilde{K}} \underbrace{(Bv)}_{u} = a \underbrace{(Bv)}_{u}$. It is clear that A must be the matrix with the eigenvalues of $\tilde{K}$. The matrix Bv is the matrix of the corresponding eigenvectors, which are orthogonal but not necessary orthonormal. The normalization can be computed from the decomposition: $\tilde{K} = B \underbrace{VV^\top}_{diag(1)} B^\top = BVA^{-1/2}AA^{-1/2}V^\top B^\top = CAC^\top$, where we defined $C = BVA^{-1/2}$ as the matrix of orthonormal eigenvectors of K and diag(1) refers to a zero matrix with 1 on all diagonal elements. The eigenvalues of $\tilde{K}$ can be obtained using $A = C^\top \tilde{K} C$. Using this derivation we can obtain exact eigenvalues and eigenvectors of an indefinite low rank kernel matrix K, given $\text{rank}(K) = m$ and the landmarks points are independent.[7]

5. Indefinite CVM

The indefinite CVM can be obtained by carefully combining the former mentioned concepts, namely the stabilization problem of the KSVM, the reformulation of the kernel as a MEB optimization problem for CVM and the Nyström approximation to ensure low computational costs.[8] As the problem solver we use a CVM solver.

The resulting algorithm, that computes the solution of the stabilization problem by solving the equivalent SVM dual minimization problem within the Core Vector Machine framework is given by Algorithm 3 and named iCVM (for indefinite CVM).

Algorithm 3 Indefinite Core Vector Machine (iCVM).

Indefinite CVM:
ζ - vector of landmarks (e.g. randomly selected)
approximate K' (the 2-class CVM kernel, Eq. (4)) using ζ as shown in Eq (5) to obtain $\tilde{K}'$
$[U, D]$ = NyströmEigenDecomposition($\tilde{K}'$) (see Sec 4)
$\hat{K} = USDU^\top$ with $S = sign(D)$ $\hat{K}$ is not generated as a full matrix - see text
$[\alpha]$ = CoreVectorMachineSolver($\hat{K}$, Y, C)
$\tilde{\alpha} = USU^\top \alpha \qquad b = Y\tilde{\alpha}^\top$
return $\tilde{\alpha}, b$;

The kernel matrix is never constructed to a $N \times N$ matrix, but we always use a Nyström approximated formulation. The values of the flipped kernel, which serves as an input of the CVM can either again be approximated by another Nyström approximation or provided more directly by using the matrices U, D of the eigendecomposition. The later one is sufficient if the kernel matrix $\hat{K}$ is

[5] In very rare cases - e.g. if multiple columns and rows in the kernel matrix $\hat{K}$ are identical it could happen that the core set optimization problem gets ill-posed. In this case the last valid (or initial) sub-optimal solution can be used. Similar numerical problems may also happen for a classical SVM.

[6] It is exact, if our low rank assumption holds. In each case the costs are linear.

[7] An implementation of this approach is available at http://techfak.uni-bielefeld.de/~bmokbel/published_code/Nystroem_toolbox.zip provided from [34].

[8] With a full kernel the costs exceed $\mathcal{O}(N^2)$ and it is useless to apply CVM instead of KSVM.

not evaluated to often which is the case for CVM. Note that the bias parameter is calculated using $\tilde{\alpha}$.

The Algorithm 3 solves[9] the complexity issue for KSVM by providing a linear cost strategy instead of quadratic to cubic complexity of KSVM.[10] However, the obtained parameters $\tilde{\alpha}$ are still dense, an issue which is addressed in the next section.

5.1. Sparsification of iCVM

The parameters $\tilde{\alpha}$ are dense as already noticed in [2]. A naive sparsification by using only $\tilde{\alpha}_i$ with large absolute magnitude is not possible as can be easily checked by counter examples. Also classical strategies such as orthogonal matching pursuit used in [44] do not work well in general. We suggest to restrict the projection operator and hence the transformation matrix of iCVM to a subset of the original training data. To get a consistent solution we have to recalculate parts of the eigen-decomposition as shown in Algorithm 4. To obtain the respective subset of the training data we use the samples which are core vectors.[11] The number of core vectors is guaranteed to be very small [39] and hence even for a larger number of classes the solution remains widely sparse. The suggested approach is given in Algorithm 4. We assume that the original projection function (line $f(\tilde{\alpha}) = USU^\top\alpha$ of Algorithm 4, detailed in [2]), is smooth and can be potentially restricted to a small number of construction points with low error. As shown in the experiments this sparsification works well very often, but we have also datasets where the smoothness assumption does not hold. In these cases the error rate increases by a significant amount see e.g. for the swissprot data. A more detailed analysis reveals that this is typically the case for datasets with a high intrinsic dimensionality or a large amount of non-vanishing eigenvalues, respectively. Apparently this is a property of the corresponding data set and not a failure of the method. It should be noted that if the input kernel K was already approximated by a Nyström approximation an out of sample extension to potentially all training points can be easily done by using the Nyström kernel expansion [23], hence sparsification of the KSVM or iCVM models is not very severe in such cases, but helpful to further reduce the computational costs in the test phase.[12]

Algorithm 4 Sparsification of iCVM.

Sparse iCVM:
Apply iCVM from Alg. 3
ζ - vector of projection points by using the core set points
construct a reduced K' using indices ζ as $\bar{K}$
[U,D] = EigenDecomposition($\bar{K}$)
$\bar{\alpha} = USU^\top\alpha$ with $S = sign(D)$ and U restricted to the core set indices
$\tilde{\alpha} = 0 \quad \tilde{\alpha}_\zeta = \bar{\alpha}$ - map the transformed alphas to $\tilde{\alpha}$
$b = Y\tilde{\alpha}^\top$
return $\tilde{\alpha}, b$;

[9] An implementation of the iCVM and the sparse iCVM is provided at - blind for review -

[10] KSVM has sub-cubic complexity if only a few dominating eigenvalues are determined.

[11] A similar strategy for KSVM may be possible but is much more complicated because typically quite many points are support vectors and special sparse SVM solvers would be necessary.

[12] Using the Nyström kernel expansion only the proximities to the landmarks have to be calculated, the proximities to all non-vanishing $\tilde{\alpha}$ can be obtained by a simple matrix operation.

Table 1
Overview of the different datasets. We provide the dataset size (N) and the origin of the indefiniteness. For vectorial data the indefiniteness is caused artificial by using the tanh kernel. But most often it occurs due to a domain specific non-metric similarity measure.

Dataset	#samples	proximity measure and data source
CatCortex	65	cortical connexion strength [27]
Aural	100	similarity based on human perception [29]
balls_3	200	synthetic dissimilarity [30]
Protein_ksvm	213	sequence-alignment similarity [27]
Patrol	241	similarity based on human memory [29]
CoilYork	228	Graph matching [30]
Chicken15_45	446	weighted edit distance between images contours [27]
Chicken29_45	446	weighted edit distance between images contours [27]
Diabetes_tanh	768	tanh kernel [19]
Sonatas	1068	normalized compression distance on midi files [31]
Delft	1500	dynamic time warping [17]
a1a	1605	tanh kernel [19]
zongker	2000	template matching on handwritten digits [27]
prodom	2604	pairwise structural alignment on proteins [27]
PolydistH57	4000	Hausdorff distance [27]
chromo	4200	edit distance on chromosomes [27]
Mushrooms	8124	tanh kernel [32]
swiss-10k	$\approx 10k$	smith waterman alignment on protein sequences [17]
checker-100k	100.000	tanh kernel (indefinite)
skin	245.057	tanh kernel (indefinite)[33]
checker	1 Mill	tanh kernel (indefinite)

6. Experiments

This part contains a series of experiments that show that our approach leads to a substantially lower complexity, while keeping similar prediction accuracy compared to KSVM. We follow the experimental design given in [2]. We also show the best published result so far summarizing additional comparisons to alternative methods. Methods that require to modify test data are excluded as also done in [2]. Finally we compare the experimental complexity of the different solvers. The used data are explained in Table 1. Additional larger data sets have been added to motivate our approach in the line of learning with large scale indefinite kernels. Results, reported for SVM on indefinite kernel matrices are obtained by using the SimpleSVM implementation of [2]. The iCVM implementation is matlab based using the code fragments mentioned before and by employing a plain quadratic problem solver. Accordingly we do not have any heuristic stopping criteria for the solve MEB(S) step in Algorithm 2. We use the probabilistic sampling strategy as suggested in [22] for the outer loop.

6.1. Experimental setting

For each dataset, we have run 20 times the following procedure: a random split to produce a training and a testing set, a 5-fold cross validation to tune each parameter (the number of parameters depending on the method) on the training set, and the evaluation on the testing set. If $N < 1000$ we use $m = 200$ randomly chosen landmarks and $N = m$ otherwise. If the input data are vectorial data we used a tanh kernel with parameters [1, 1] to obtain an indefinite kernel.

6.2. Results

Table 2 gives average error rates and standard deviation of KSVM-L and iCVM and for comparison the best published result found in the literature is reported. We also report the results obtained by a standard SVM. We observe severe convergence

Table 2
Prediction errors on the test set - small scale indefinite kernels.

	#samples	iCVM	KSVM-L	SVM	Others
CatCortex	65	12.31 ± 6.88⋆	**5.4 ± 6.3**	72.33 ± 22.82	7.0 ± 7.1
CatCortex				no convergence	[27]
Aural	100	**12.00 ± 4.47**	12.5 ± 6.17	13.00 ± 4.47	12 ± 6
Aural					[29]
balls_3d[a]	200	**0.50 ± 1.12⋆**	41.37 ± 6.67	50.00 ± 11.59	45.70 ± 1.7
balls_3d				no convergence	[30]
Protein_ksvm	213	0.48 ± 1.06	**0.2 ± 0.7**	54.45 ± 22.29	0.4 ± 1.7
Protein_ksvm				no convergence	[27]
Patrol	241	28.63 ± 6.81⋆	12.29 ± 4.56	27.38 ± 5.15	**11.56 ± 4.54**
Patrol					[29]
CoilYork	228	36.50 ± 8.23	33.10 ± 5.05	77.77 ± 2.37	**33.6 ± 1.2**
CoilYork				no convergence	[30]
Chicken15_45	446	6.73 ± 1.38	**6.34 ± 2.45**	65.90 ± 15.09	7 ± 2.8
Chicken15_45				no convergence	[27]
Chicken29_45	446	7.63 ± 2.45	**4.6 ± 2.5**	74.00 ± 3.04	4.7 ± 2.7
Chicken29_45				no convergence	[27]
Diabetes_tanh	768	23.30 ± 4.03	**22.59 ± 2.30**	22.92 ± 2.38	22.92
Diabetes_tanh					[19]

[a] The worse results of the other methods are caused by a wrong dissimilarity to similarity transformation. If it is done correctly the error rates are comparable.

Table 3
Prediction errors on the test set - large scale indefinite kernels.

	#samples	iCVM	KSVM-L	Others
Sonatas	1068	13.01 ± 3.82	15.92 ± 2.59	**11.52 ± 0.20** [31]
Sonatas (time)	1068	77.92	57.40	–
Delft	1500	3.20 ± 0.84	11.13 ± 3.38	**1.80 ± 1.48**[17]
Delft (time)	1500	9.44	475.31	–
a1a	1605	20.56 ± 1.34	17.24 ± 1.88	**17.08** [19]
a1a (time)	1605	6.87	72.63	–
zongker	2000	6.40 ± 2.11	8.75 ± 0.68	**4.4 ± 0.6** [27]
zongker (time)	2000	8.82	395.34	–
prodom	2604	**0.87 ± 0.64**	0.9 ± 0.3	1.3 ± 0.5 [27]
prodom (time)	2604	25.63	2341.80	–
PolydistH57	4000	**0.70 ± 0.19**	1.86 ± 0.50	5.4 ± 1.3 [27]
PolydistH57 (time)	4000	5.65	9990.30	–
chromo	4200	6.10 ± 0.63	**5.3 ± 0.3**	7.7 ± 0.4 [27]
chromo (time)	4200	35.68	11563.30	–
Mushrooms	8124	2.54 ± 0.56	5.08 ± 0.73	**1.09** [32]
Mushrooms (time)	8124	45.19	65225.50	–
swiss-10k	10998*k*	12.08 ± 3.47⋆	n.a.	**1.41 ± 0.35** [17]
swiss-10k (time)	10998	73.72	n.a.	n.a.
checker-100k	100.000	**9.66 ± 2.32**	n.a.	n.a.
checker-100k (time)	100.000	112.62	n.a.	n.a.
skin	245.057	**4.22 ± 1.11**	n.a.	n.a.
skin (time)	245.057	258.62	n.a.	n.a.
checker	1 Mill	**9.38 ± 2.73**	n.a.	n.a.
checker (time)	1 Mill	1212.21	n.a.	n.a.

problems for almost all datasets using a classical SVM.[13] Only for Aural, Patrol and Diabetes_tanh the SVM training converged, with acceptable error rates on the test data. In most cases the SVM model has a very high error rate for the given data - clearly indicating a need for an adapted training procedure or algorithm. We observe that KSVM-L is always more accurate or close to the best published result and that iCVM is very close to KSVM-L. While we are not expecting any improvements of iCVM over KSVM-L we sometimes see also better results. This maybe advocated to the improved representation accuracy of the kernel matrix in iCVM in contrast to KSVM-L. In [2] for KSVM-L only the top dominating eigenvalues are used to keep the computational load tractable, while in our approach we can be more flexible due to the Nyström approximated eigen-decomposition. iCVM performs in general better for larger datasets because the approximations e.g. the ϵ-ball approximation introduces additional errors for small scale data sets. In Table 3 we summarize results for larger scale data which in parts could not be any longer processed by KSVM-L (and not at all by KSVM), due to the large number of samples. Significant differences of iCVM to the best result are indicated by a $\star$ (anova, $p < 5\%$). Again, the iCVM is similar accurate compared with KSVM-L or alternative results. We also report runtime results. While for the smaller datasets the runtimes of iCVM and KSVM-L are similar the iCVM is substantially faster with linear instead of quadratic complexity for larger datasets. In Table 4 we show the results for large scale data (having at least 1000 points) using iCVM with sparsification. We observe much smaller models, especially for larger datasets with often comparable prediction accuracy with respect to the non-sparse model. The runtimes are similar to the non-sparse case but in general slightly higher due to the extra eigen-decompositions on a reduce set of the data as shown in Algorithm 4.

[13] The same also happens for the larger data sets but due to the SVM convergence problems with long runtimes we skipped these experiments.

Table 4
Prediction errors on the test set for large scale indefinite kernels with a sparse mapping. The percentage of projection points is calculated using the unique set over core vectors over all classes in comparison to all training points. Datasets with substantially reduced prediction accuracy are marked by ⊙.

	#samples	iCVM (sparse)	projection pts	iCVM (non-sparse)
Sonatas	1068	12.64 ± 1.71	76.84%	13.01 ± 3.82
Sonatas (time)	1068	269.31	–	77.92
Delft	1500	16.53 ± 2.79⊙	52.48%	3.20 ± 0.84
Delft (time)	1500	12.35	–	9.44
a1a	1605	39.50 ± 2.88⊙	1.25%	20.56 ± 1.34
a1a (time)	1605	21.53	–	6.87
zongker	2000	29.20 ± 2.48⊙	52.81%	6.40 ± 2.11
zongker (time)	2000	42.19	–	8.82
prodom	2604	2.89 ± 1.17	26.31%	0.87 ± 0.64
prodom (time)	2604	35.75	–	25.63
PolydistH57	4000	6.12 ± 1.38	12.92%	0.70 ± 0.19
PolydistH57 (time)	4000	31.72	–	5.65
chromo	4200	11.50 ± 1.17	33.76%	6.10 ± 0.63
chromo (time)	4200	37.05	–	35.68
Mushrooms	8124	7.84 ± 2.21	6.46%	2.54 ± 0.56
Mushrooms (time)	8124	59.89	–	45.19
swiss-10k	≈ 10*k*	35.90 ± 2.52⊙	17.03%	12.08 ± 3.47
swiss-10k (time)	10998	214.98	–	73.72
checker-100k	100.000	8.54 ± 2.35	2.26%	9.66 ± 2.32
checker-100k (time)	100.000	179.02	–	112.62
skin	245.057	9.38 ± 3.30	0.06%	4.22 ± 1.11
skin (time)	245.057	234.53	–	258.62
checker	1 Mill	8.94 ± 0.84	0.24%	9.38 ± 2.73
checker (time)	1 Mill	1736.21	–	1212.21

6.3. Complexity analysis

The original KSVM has runtime costs (with full eigen-decomposition) of $\mathcal{O}(N^3)$ and memory storage $\mathcal{O}(N^2)$, where N is the number of points. The iCVM involves the extra Nyström approximation of the kernel matrix to obtain $K_{(N,\,m)}$ and $K^{-1}_{(m,m)}$, if not already given. If we have m landmarks, $m \ll N$, this gives memory costs of $\mathcal{O}(mN)$ for the first matrix and $\mathcal{O}(m^3)$ for the second, due to the matrix inversion. Further a Nyström approximated eigendecomposition has to be done to apply the eigenspectrum flipping operator. This leads to runtime costs of $\mathcal{O}(N \times m^2)$. The runtime costs for the sparse iCVM are $\mathcal{O}(N \times m^2)$ and the memory complexity is the same as for iCVM. Due to the used Nyström approximation the prior costs only hold if $m \ll N$, which is the case for many datasets as shown in the experiments.

The application of a new point to a KSVM or iCVM model requires the calculation of kernel similarities to all N training points, for the sparse iCVM this holds only in the worst case. In general the sparse iCVM provides a simpler out of sample extension as shown in Table 4, but is data dependent.

The (i)CVM model generation has not more than N iterations or even a constant number of 59 points, if the probabilistic sampling trick is used [45]. As show in [39] the classical CVM has runtime costs of $\mathcal{O}(1/\epsilon^2)$. The evaluation of a kernel function using the Nyström approximated kernel can be done with cost of $O(m^2)$ in contrast to constant costs if the full kernel is available. Accordingly, If we assume $m \ll N$ the overall runtime and memory complexity of iCVM is linear in N, this is two magnitudes less as for KSVM for reasonable large N and for low rank input kernels.

Fig. 1 shows in log scales the training and testing time together depending on the training set size. The test set size is constant to 1000. The experiment shows that iCVM is substantially faster than KSVM-L (KSVM using partial eigen-decomposition).

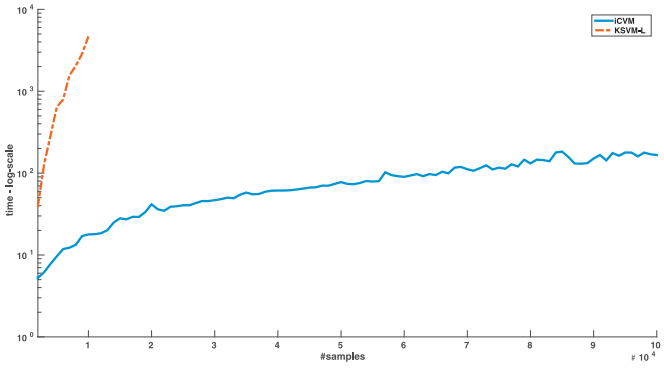

Fig. 1. Runtime of the checkerboard data (with an indefinite tanh kernel) with 1000 – 100.000 points (x-axis). The straight line indicates the iCVM and the dotted line the KSVM-L results. Runtime is plotted on the y-axis at log-scale. One can clearly sees a linear complexity of $\mathcal{O}(N)$ for iCVM and a roughly quadratic complexity for KSVM-L.

7. Discussions and conclusions

As discussed in [2], there is no good reason to enforce positive-definiteness in kernel methods. A very detailed discussion on reasons for using KSVM or now iCVM is given in [2], explaining why a number of alternatives or pre-processing techniques are in general inappropriate. Our experimental results show that an appropriate Krĕin space model is not only at least as effective as other approaches but it does consistently so, across a number of datasets. Although the original learning methods in Krĕin spaces can be costly, the presented approach provides an algorithm with linear complexity if the input kernel is low rank. While not all input kernels may be of low rank, empirical experiments showed that this assumption holds often in practice or can be imposed without severe negative impact on the prediction accuracy. As is the case for KSVM, the presented approach can be applied without the need for transformation of test points, which is a desirable property for practical applications.

Acknowledgment

A Marie Curie Intra-European Fellowship (IEF): FP7-PEOPLE-2012-IEF (FP7-327791-ProMoS) and support from the Cluster of Ex-

cellence 277 Cognitive Interaction Technology funded by the German Excellence Initiative is gratefully acknowledged. PT was supported by the EPSRC grant EP/L000296/1, Personalized Health Care through Learning in the Model Space. We would like to thank R. Duin, Delft University for various support with distools and prtools and Gaelle Bonnet-Loosli for providing support with the Krĕin Space SVM.

References

[1] V. Vapnik, The nature of statistical learning theory, Statistics for Engineering and Information Science, Springer, 2000.
[2] G. Loosli, S. Canu, C.S. Ong, Learning SVM in Krein spaces, IEEE Trans. Pattern Anal. Mach. Intell. 38 (6) (2016) 1204–1216.
[3] C. Hodgetts, U. Hahn, Similarity-based asymmetries in perceptual matching, Acta Psychol. 139 (2) (2012) 291–299.
[4] W. Xu, R. Wilson, E. Hancock, Determining the cause of negative dissimilarity eigenvalues, in: Lecture Notes in Computer Science, in: 6854 LNCS (PART 1), 2011, pp. 589–597.
[5] L. Van Der Maaten, G. Hinton, Visualizing non-metric similarities in multiple maps, Mach. Learn. 87 (1) (2012) 33–55.
[6] W.J. Scheirer, M.J. Wilber, M. Eckmann, T.E. Boult, Good recognition is non–metric, Pattern Recognit. 47 (8) (2014) 2721–2731.
[7] E. Pekalska, R.P.W. Duin, S. Günter, H. Bunke, On not making dissimilarities euclidean, in: Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18–20, 2004 Proceedings, 2004, pp. 1145–1154.
[8] R.P.W. Duin, E. Pekalska, Non-euclidean dissimilarities: causes and informativeness, in: Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR&SPR 2010, Cesme, Izmir, Turkey, August 18–20, 2010. Proceedings, 2010, pp. 324–333.
[9] A. Cichocki, S.-I. Amari, Families of alpha- beta- and gamma- divergences: flexible and robust measures of similarities, Entropy 12 (6) (2010) 1532–1568.
[10] Z. Zhang, B.C. Ooi, S. Parthasarathy, A.K.H. Tung, Similarity search on Bregman divergence: towards non-metric indexing, Proc. VLDB Endow. 2 (1) (2009) 13–24.
[11] D. Schnitzer, A. Flexer, G. Widmer, A fast audio similarity retrieval method for millions of music tracks, Multimedia Tools Appl. 58 (1) (2012) 23–40.
[12] E. Mwebaze, P. Schneider, F.-M. Schleif, J. Aduwo, J. Quinn, S. Haase, T. Villmann, M. Biehl, Divergence based classification in learning vector quantization, Neurocomputing 74 (2010) 1429–1435.
[13] F. van der Meer, The effectiveness of spectral similarity measures for the analysis of hyperspectral imagery, Int. J. Appl. Earth Obs. Geoinf. 8 (1) (2006) 3–17.
[14] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, Acoustics, Speech Signal Process. IEEE Trans. 26 (1) (1978) 43–49.
[15] H. Ling, D.W. Jacobs, Shape classification using the inner-distance, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 286–299.
[16] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.
[17] F. Schleif, P. Tiňo, Indefinite proximity learning: a review, Neural Comput. 27 (10) (2015) 2039–2096.
[18] B. Haasdonk, Feature space interpretation of SVMS with indefinite kernels, IEEE Trans. Pattern Anal. Mach. Intell. 27 (4) (2005) 482–492.
[19] R. Luss, A. d'Aspremont, Support vector machine classification with indefinite kernels, Math. Program. Comput. 1 (2–3) (2009) 97–118.
[20] S. Gu, Y. Guo, Learning SVM classifiers with indefinite kernels, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22–26, 2012, 2012.
[21] A. Munoz, I. De Diego, From indefinite to positive semi-definite matrices, Lecture Notes in Computer Science4109 LNCS (2006) 764–772.
[22] I.W. Tsang, J.T. Kwok, P. Cheung, Core vector machines: fast SVM training on very large data sets, J. Mach. Learn. Res. 6 (2005) 363–392.
[23] C.K.I. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems 13: NIPS'2000, 2000, pp. 682–688.
[24] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences., J. Mol. Biol. 147 (1) (1981) 195–197.
[25] M. Neuhaus, H. Bunke, Edit distance based kernel functions for structural pattern classification, Pattern Recognit. 39 (10) (2006) 1852–1863.
[26] B. Haasdonk, E. Pekalska, Indefinite kernel fisher discriminant, in: 19th International Conference on Pattern Recognition (ICPR 2008), December 8–11, 2008, Tampa, Florida, USA, IEEE Computer Society, 2008, pp. 1–4.
[27] E. Pekalska, B. Haasdonk, Kernel discriminant analysis for positive definite and indefinite kernels, IEEE Trans. Pattern Anal. Mach. Intell. 31 (6) (2009) 1017–1031.
[28] H. Chen, P. Tino, X. Yao, Probabilistic classification vector machines, IEEE Trans. Neural Netw. 20 (6) (2009) 901–914.
[29] Y. Chen, M.R. Gupta, Fusing similarities and kernels for classification, in: 12th International Conference on Information Fusion, FUSION '09, Seattle, Washington, USA, July 6–9, 2009, 2009, pp. 474–481.
[30] B. Duin, E. Pekalska, Beyond features: similarity-based pattern analysis and recognition (simbad, eu project deliverable), 2009.
[31] B. Hammer, D. Hofmann, F. Schleif, X. Zhu, Learning vector quantization for (dis-)similarities, Neurocomputing 131 (2014) 43–51.
[32] A. Srisuphab, J. Mitrpanont, Gaussian kernel approximation algorithm for feedforward neural network design, Appl. Math. Comput. 215 (7) (2009) 2686–2693.
[33] UCI, Skin segmentation database, 2016 http://archive.ics.uci.edu/ml/datasets/Skin+Segmentation.
[34] A. Gisbrecht, F. Schleif, Metric and non-metric proximity transformations at linear costs, Neurocomputing 167 (2015) 643–657.
[35] E. Pekalska, R. Duin, The Dissimilarity Representation for Pattern Recognition, World Scientific, 2005.
[36] C.S. Ong, X. Mary, S. Canu, A.J. Smola, Learning with non-positive kernels, in: C.E. Brodley (Ed.), Machine Learning, Proceedings of the 21st International Conference on (ICML 2004), ACM International Conference Proceeding Series, 69, ACM, 2004.
[37] J. Bognár, Indefinite inner product spaces, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer, 1974.
[38] B. Hassibi, Indefinite metric spaces in estimation, control and adaptive filtering, Stanford University, Dept. of Electrical Engineering, Stanford, 1996 Ph.D. thesis.
[39] I.W.-H. Tsang, J.T.-Y. Kwok, J.M. Zurada, Generalized core vector machines, IEEE Trans. Neural Netw. 17 (5) (2006) 1126–1140.
[40] G. Loosli, S. Canu, Comments on the "core vector machines: fast SVM training on very large data sets", J. Mach. Learn. Res. 8 (2007) 291–301.
[41] M. Badoiu, K.L. Clarkson, Optimal core-sets for balls, Comput. Geom. 40 (1) (2008) 14–22.
[42] P. Drineas, M. Magdon-Ismail, M.W. Mahoney, D.P. Woodruff, Fast approximation of matrix coherence and statistical leverage, J. Mach. Learn. Res. 13 (2012) 3475–3506.
[43] C. Hsieh, S. Si, I.S. Dhillon, Fast prediction for large-scale kernel machines, in: Advances in Neural Information Processing Systems 27: NIPS'2014, 2014, pp. 3689–3697.
[44] D. Hofmann, F.-M. Schleif, B. Hammer, Learning interpretable kernelized prototype-based models, Neurocomputing 131 (2014) 43–51.
[45] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: P. Langley (Ed.), Proceedings of the 17th International Conference on Machine Learning (ICML 2000), Morgan Kaufmann, 2000, pp. 911–918.

Frank-Michael Schleif (Dipl.-Inf, University of Leipzig, Ph.D., TU-Clausthal, Germany) was a Marie Curie Senior Research Fellow at the University of Birmingham, Birmingham, UK and a Post-Doctoral Fellow in the group of Theoretical Computer Science (TCS) at the University of Bielefeld, Bielefeld, Germany, where he also received a venia legendi in applied computer science in 2013. He was also a software developer and consultant for the Bruker Corp. Since 2016 he is with the University of Applied Sciences, Wuerzburg, Germany, where he is a Professor for Database Management and Business Intelligence. His current research interests include data management, computational intelligence techniques and machine learning for non-metric models and large scale problems. Several research stays have taken him to UK, the Netherlands, Japan and the USA. He is a member of the German chapter of the European Neural Network Society (GNNS), the GI and the IEEE-CIS. He is editor of the Machine Learning Reports and member of the editorial board of the Neural Processing Letters.

Peter Tino (M.Sc. Slovak University of Technology, Ph. D. Slovak Academy of Sciences) was a Fulbright Fellow with the NEC Research Institute, Princeton, NJ, USA, and a Post-Doctoral Fellow with the Austrian Research Institute for AI, Vienna, Austria, and with Aston University, Birmingham, UK. Since 2003, he has been with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham, UK, where he is currently a Full Professor-Chair in Complex and Adaptive Systems. His current research interests include dynamical systems, machine learning, probabilistic modelling of structured data, evolutionary computation, and fractal analysis. Peter was a recipient of the Fulbright Fellowship in 1994, the U.K.-Hong-Kong Fellowship for Excellence in 2008, three Outstanding Paper of the Year Awards from the IEEE Transactions on Neural Networks in 1998 and 2011 and the IEEE Transactions on Evolutionary Computation in 2010, and the Best Paper Award at ICANN 2002. He serves on the editorial boards of several journals.