Complex-Valued Embeddings of Generic Proximity Data

Maximilian Münch[1,2(✉)], Michiel Straat[2], Michael Biehl[2], and Frank-Michael Schleif[1]

[1] Department of Computer Science and Business Information Systems, University of Applied Sciences Würzburg-Schweinfurt, 97074 Würzburg, Germany
{maximilian.muench,frank-michael.schleif}@fhws.de

[2] Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands
{m.j.c.straat,m.biehl}@rug.nl

Abstract. Proximities are at the heart of almost all machine learning methods. In a more generic view, objects are compared by a (symmetric) similarity or dissimilarity measure, which may not obey particular mathematical properties. This renders many machine learning methods invalid, leading to convergence problems and the loss of generalization behavior. In many cases, the preferred dissimilarity measure is not metric. If the input data are non-vectorial, like text sequences, proximity-based learning is used or embedding techniques can be applied. Standard embeddings lead to the desired fixed-length vector encoding, but are costly and are limited in preserving the full information. As an information preserving alternative, we propose a complex-valued vector embedding of proximity data, to be used in respective learning approaches. In particular, we address supervised learning and use extensions of prototype-based learning. The proposed approach is evaluated on a variety of standard benchmarks showing good performance compared to traditional techniques in processing non-metric or non-psd proximity data.

Keywords: Proximity learning · Embedding · Complex valued data · Learning vector quantizer · Krein space

1 Introduction

Machine learning has a growing impact in various fields and the considered input data become more and more generic [9,12]. In particular non-vectorial data like

MM is supported by the ESF (WiT-HuB 4/2014–2020), project KI-trifft-KMU, StMBW-W-IX.4-6-190065. M.B. and M.S. acknowledge support through the Northern Netherlands Region of Smart Factories (RoSF) consortium, lead by Noordelijke Ontwikkelings en Investerings Maatschappij (NOM), The Netherlands, see http://www.rosf.nl.

A. Torsello et al. (Eds.): S+SSPR 2020, LNCS 12644, pp. 14–23, 2021.
https://doi.org/10.1007/978-3-030-73973-7_2

text data, biological sequence data, graphs, and other input formats are used [14]. The vast majority of learning algorithms expect fixed-length real value vector data as inputs and can not directly be used on non-standard data [12].

Using embedding approaches is one strategy to obtain a vectorial embedding, but this is costly, needs large amounts of data to train the embedding and information is only partially preserved [12]. In a more generic scenario, proximity measures, like alignment functions, can be applied to compare non-vectorial objects to obtain a proximity score between two objects. If all N input objects are pairwise compared, we obtain a proximity matrix $P \in \mathbf{R}^{N \times N}$. If the measure is a metric dissimilarity measure, we have a distance matrix, which can be used for the nearest-mean classifier. In the case of inner products like the Euclidean inner product, a kernel matrix is obtained. If this kernel matrix is positive semidefinite (psd), multiple kernel methods can be used [16]. Also, so-called empirical feature space approaches have been considered, but with the drawback of high model complexity and inherent data transformations [7].

Here we consider non-vectorial input data given either by a non-metric dissimilarity measure or a non-standard inner product, leading to an indefinite kernel function. As detailed in [14], learning models can be calculated on these generic proximity data in very different ways. Most often, the proximities are transformed to fit into classical machine learning algorithms, with a number of limitations [14]. In this work, we propose the application of a complex-valued embedding on these data to overcome some of the limitations. Recently different classical learning algorithms have been extended to complex-valued inputs [18]. It is now possible to preserve the information provided in the generic proximity data while learning in a fixed-length vector space using a highly effective, well-understood learning algorithm. The respective procedures are detailed in the following and evaluated on classical benchmark data with strong results.

2 Background and Basic Notation

Consider a collection of N objects $\mathbf{x}_i$, $i = \{1, 2, ..., N\}$, in some input space $\mathcal{X}$. Given a similarity function or inner product on $\mathcal{X}$, corresponding to a metric, one can construct a Mercer kernel acting on pairs of points from $\mathcal{X}$. For example, if $\mathcal{X}$ is a finite-dimensional vector space, a classical similarity function in this space is the Euclidean inner product (corresponding to the Euclidean distance).

2.1 Positive Definite Kernels - Hilbert Space

The Euclidean inner product is also known as linear kernel with $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, where ϕ is the identity mapping. Another prominent kernel function is $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma^2}\right)$, with $\sigma > 0$ as a free scale parameter. In any case, it is assumed that the kernel function $k(\mathbf{x}, \mathbf{x}')$ is psd.

The transformation ϕ is, in general, a *non-linear* mapping to a high-dimensional Hilbert space $\mathcal{H}$ and may not be given in an explicit form, but

allowing *linear* techniques in $\mathcal{H}$. Instead of providing an explicit mapping, a kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is given, which encodes the inner product in $\mathcal{H}$. The kernel k is a positive (semi-) definite function such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$, for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. The matrix $K_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j)$ is an $N \times N$ kernel (Gram) matrix derived from the data. For general similarity measures, we use $\mathbf{S}$ to describe the respective similarity matrix.

Kernelized methods process the embedded data points in a feature space utilizing only the inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ [16], without the need to explicitly calculate ϕ, known as *kernel trick*. Explicit mappings of psd kernel function are also frequently used to employ linear methods. However, the underlying similarity function may not be metric, but a domain-specific similarity measure, as mentioned before. Such similarity measures imply *indefinite* kernels, preventing standard "kernel-trick" methods developed for Mercer kernels to be applied.

2.2 Non-positive Definite Kernels - Krein Space

A Krein space is an *indefinite* inner product space endowed with a Hilbertian topology. 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: $\langle \alpha f + g, h \rangle_{\mathcal{K}} = \alpha \langle f, h \rangle_{\mathcal{K}} + \langle g, h \rangle_{\mathcal{K}}$;
- $\langle f, g \rangle_{\mathcal{K}} = 0$ implies $f = 0$.

A vector space $\mathcal{K}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{K}}$ is called an inner product space. An inner product space $(\mathcal{K}, \langle \cdot, \cdot \rangle_{\mathcal{K}})$ is a Krein space if we have two Hilbert spaces $\mathcal{H}_+$ and $\mathcal{H}_-$ spanning $\mathcal{K}$ such that $\forall f \in \mathcal{K}$ we have $f = f_+ + f_-$ with $f_+ \in \mathcal{H}_+$ and $f_- \in \mathcal{H}_-$ and $\forall f, g \in \mathcal{K}$, $\langle f, g \rangle_{\mathcal{K}} = \langle f_+, g_+ \rangle_{\mathcal{H}_+} - \langle f_-, g_- \rangle_{\mathcal{H}_-}$.

Indefinite kernels are typically observed by means of domain-specific non-metric similarity functions (such as alignment functions used in biology [17]), by specific kernel functions - e.g., the Manhattan kernel $k(x, x') = -||x - x'||_1$ or others. A finite-dimensional Krein-space is a so-called pseudo-Euclidean space.

3 Embedding for Non-psd Proximities

Embedding of a proximity matrix into a vector space is not a new consideration, see e.g. [5], but was shown to be valid so far only in case of psd kernel functions. Given a symmetric *dissimilarity* matrix with zero diagonal, an embedding of the data in a pseudo-Euclidean vector space, determined by the eigenvector decomposition of the associated similarity matrix $\mathbf{S}$, is always possible [3][1]. Given the

[1] The associated similarity matrix can be obtained by double centering [12] of the dissimilarity matrix. $\mathbf{S} = -\mathbf{JDJ}/2$ with $\mathbf{J} = (\mathbf{I} - \mathbf{1}\mathbf{1}^\top/N)$, identity matrix $\mathbf{I}$ and vector of ones $\mathbf{1}$.

eigendecomposition of $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, we can compute the corresponding vectorial representation $\mathbf{V}$ in the pseudo-Euclidean space by

$$\mathbf{V} = \mathbf{U}_{p+q+z} \, |\mathbf{\Lambda}_{p+q+z}|^{1/2} \tag{1}$$

where $\mathbf{\Lambda}_{p+q+z}$ consists of p positive, q negative non-zero eigenvalues, and z zero eigenvalues. $\mathbf{U}_{p+q+z}$ consists of the corresponding eigenvectors. The triplet (p, q, z) is also referred to as the signature of the pseudo-Euclidean space. The crucial point in Eq. (1) is the *absolute* operator used in the embedding, which is also called a flip operation in the field of indefinite learning [14]. This very costly operation makes the data metric and alters the underlying data structure [9].

The transformation of dissimilarities to obey metric properties, or of similarities to be psd is at least technically useful because it permits to employ many mathematical concepts [5], not available otherwise. We remove the absolute function from the embedding in Eq. (1) and obtain Eq. (2), and show later how the embedding can be made computational effective also for non-psd inputs. Apparently, the new embedding does not modify the data, in particular, an inner product of the embedded data reveals the input's information again.

$$\mathbf{V} = \mathbf{U}_{p+q+z}\mathbf{\Lambda}_{p+q+z}^{1/2} \tag{2}$$

The real-valued embedding in Eq. (1), leading to a psd formulation, and the complex-valued embedding in Eq. (2) is straight forward but extremely costly. Already in [5], the costs in Eq. (1) were addressed by using the Nyström approximation, applicable to the psd case only. This approach can not be used directly in our setting since the input is non-psd.

In our former work [2] (simplified in [11]), we have shown that the Nyström approximation remains valid for generic proximity data, in particular non-psd similarities. Hence the Nyström approximation becomes available to approximate a non-psd matrix. In [2], we have further shown how the Nyström approximation can also be used to have an approximated Double-Centering for dissimilarity data. Our work helps twofold to permit an effective embedding in Eq. (2):

1. the input needs not to be a kernel but can also be a dissimilarity matrix
2. the Nyström matrix approximation can also be applied on non-psd similarities which reduced the costs of the embedding

In the Nyström approximation, we have to specify the number m of landmarks with $m \ll N$. The landmarks can be selected for non-psd matrices randomly or by kmeans++ as shown in [11]. Our efficient approach to get an approximated complex-valued, vectorial embedding of a non-psd matrix is shown in Algorithm 1.

In the first step of Algorithm 1, the input matrix is approximated using the Nyström approximation (potentially with an integrated double centering). this can be done with linear costs and with guaranteed approximation bounds [2,11]. Subsequently, we calculate the essential part of the embedding function in Eq. (2) combined with the projection matrix of the Nyström approximation, by

Algorithm 1. Complex valued embedding of non-psd proximities

Embed_proximities(P, m)
if P is dissimilarity **then**
  $Knm, Kmm :=$ ApproximatedDoubleCentering(P, m) using [2] and kmeans++
else
  $Knm, Kmm :=$ Approximate(P, m) using [2] for similarities and kmeans++
end if
$[C, A] :=$ eig(Kmm); with eigenvectors C and eigenvalues in A (diagonal)
$W :=$ diag(sqrt(1./diag(A))) $\cdot$ C' complex-valued Nyström projection matrix
$M :=$ W $\cdot$ Knm' complex-valued embedding
$K^* :=$ M' $\cdot$ M reconstruction (optional)
return M

taking the square root of the (pseudo-) inverse of the eigenvalue decomposition of Kmm, done with linear costs. Details to costs and approximation procedure are shown in [2]. The vectorial embedding M is finally done by mapping the rectangular Nyström part Knm of the similarities to the projection matrix W[2] If the similarity matrix K is non-psd, A contains negative eigenvalues and the embedding becomes complex-valued.

We now have an approximated complex-valued fixed-length vectorial embedding of the proximity data P, whereby the respective reconstruction is exact if the rank of P equals to the number of non-vanishing eigenvalues in A. Algorithm 1 has a linear complexity (without K^*) as long as the number of landmarks $m \ll N$, which is, in general, the case. The embedding procedure has a straight forward out of sample extension. The mapping in Algorithm 1 can be done for new points by evaluating the proximity function for the landmark point and using the respective projection function.

For the complex-valued embedding (so far), only a limited number of machine learning algorithms is available, like the complex-valued support vector machine (cSVM) [22], the complex-valued generalized learning vector quantization (cGMLVQ) [18], or a complex-valued neural network (cNNet) [19]. Further, a nearest neighbor (NN) classifier can be used by employing a standard norm operator. While cSVM, cGMLVQ, cNNet are parametric methods, the NN classifier is parameter-free and can be used directly. In particular, after applying the norm, the obtained dissimilarity values are metric. Due to its good performance and simplicity, we focus on cGMLVQ, briefly reviewed in the following.

4 Complex-Valued Generalized Learning Vector Quantization

In Learning Vector Quantization (LVQ), the classification scheme is parameterized by a set of labeled prototypes and a distance measure $d(\cdot, \cdot)$. New data is classified according to the nearest prototype's label with respect to the distance

[2] Some heuristic ideas on Landmark MDS, which is imprecise, are discussed in [2].

measure $d(\cdot,\cdot)$. In contrast to the NN classifier in which the full dataset is used, the classes in LVQ schemes are represented by only very few prototypes. Hence, in the algorithm's working phase, LVQ methods require less computational effort and storage. Moreover, LVQ is often praised for its white-box character, which is beneficial in many applications [20].

4.1 Training an LVQ Classifier

Given a training dataset of N labeled inputs $(\mathbf{x}_i, y_i)_{i=1}^N$, in which $\mathbf{x}_i \in \mathcal{R}^d$ is an input vector and $y_i \in \{1, 2, ..., K\}$ its class label. The aim of the training procedure is the adaptation of M labeled prototypes $\{(\mathbf{w}_k, y_k)\}_{k=1}^M$, such that the resulting classification scheme gives high classification accuracy with respect to unseen data. The distance measure $d(\cdot,\cdot)$ is of central importance in the training- and classification procedure. A common choice is squared Euclidean distance measure $(\mathbf{x}-\mathbf{w})^T(\mathbf{x}-\mathbf{w})$. In [13], a valid cost function for the LVQ heuristic was proposed, that can be minimized by, e.g., gradient descent:

$$E_{GLVQ} = \sum_{i=1}^{N} \Phi(\mu_i), \text{ with } \mu_i = \frac{d_+(\mathbf{x}_i) - d_-(\mathbf{x}_i)}{d_+(\mathbf{x}_i) + d_-(\mathbf{x}_i)}\,. \tag{3}$$

The argument μ_i is based on the difference between the distance $d_+(\mathbf{x}_i)$ from its position to the closest prototype with the same label and the distance $d_-(\mathbf{x}_i)$ to the closest prototype with a different label, normalized to the range $\mu_i \in [-1, 1]$. The function $\Phi(\cdot)$ is monotonically increasing and is usually chosen to be identity $\Phi(x) = x$ or the logistic function $\Phi(x) = 1/(1+\exp(-x))$. The standard Euclidean distance does not account for differences in the classification importance of the dimensions. To improve classification accuracy, matrix *relevance learning* was introduced [15]. A full matrix of adaptive relevances $\mathbf{\Lambda} = \mathbf{\Omega}^T\mathbf{\Omega}$ is introduced in the distance measure:

$$d^{\Lambda}(\mathbf{w}, \mathbf{x}_i) = (\mathbf{x}_i - \mathbf{w})^T \mathbf{\Omega}^T \mathbf{\Omega} (\mathbf{x}_i - \mathbf{w})\,, \tag{4}$$

The linear projection defined by the matrix $\mathbf{\Omega}$ is adapted during training to reflect the importance of the features and to account for correlations between features.

The above cost function in Eq. (3) is minimized with respect to the prototypes $\{\mathbf{w}_k\}_{k=1}^M$ and the linear projection matrix $\mathbf{\Omega}$ by either batch- or stochastic gradient descent. To formulate the gradient descent update rules with respect to $\mathbf{w}_\pm$ and $\mathbf{\Omega}$ for an example $\mathbf{x}_i$, one applies the chain rule:

$$\mathbf{w}_\pm = \mathbf{w}_\pm - \alpha \Phi'(\mu_i) \frac{\partial \mu_i}{\partial d_\pm} \frac{\partial d_\pm}{\partial \mathbf{w}_\pm}\,, \quad \mathbf{\Omega}_\pm = \mathbf{\Omega}_\pm - \beta \Phi'(\mu_i) \frac{\partial \mu_i}{\partial d_\pm} \frac{\partial d_\pm}{\partial \mathbf{\Omega}_\pm} \tag{5}$$

with the learning rates α and β. For all results reported in the following, we have set $\alpha = 0.01$ and $\beta = 0.001$.

4.2 Learning Rules for Complex-Valued Data

When the data lives in the complex-valued space $\mathbb{C}^d$ and one uses the Hermitian transpose in Eq. (4), the distance is always real-valued, since it is a sum of squared magnitudes. Hence, only the innermost derivatives of the distance measure in Eq. (5) have to be considered with respect to the complex-valued variables. These can be done using the Wirtinger differential operators [21] as proposed in [18]:

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial z^*} = \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right), \tag{6}$$

in which $z = x + iy$ and $z^* = x - iy$, the complex conjugate. Using the differential operator with respect to z^*, the inner most derivatives in Eq. (5) are as follows:

$$\frac{\partial d}{\partial \mathbf{w}_\pm^*} = -\mathbf{\Omega}^H\mathbf{\Omega}(\mathbf{x}_i - \mathbf{w}_\pm), \quad \frac{\partial d}{\partial \mathbf{\Omega}^*} = \mathbf{\Omega}(\mathbf{x}_i - \mathbf{w}_\pm)(\mathbf{x}_i - \mathbf{w}_\pm)^H, \tag{7}$$

which are conceptually similar to the derivatives of the real-valued variables. The model update is implicitly done in a Krein space, while the predictions are guided by the metric dissimilarities from the employed norm operator. The cG(M)LVQ model can be trained with linear costs on the vectorial data.

5 Experiments

In this section, we show the effectiveness of the proposed embedding approach on a set of benchmark data typically used in the area of proximity-based supervised learning and by employing appropriate classified models. The following section contains a brief description of the datasets with details in the references. Subsequently, we evaluate the performance of our embedding approach on these datasets compared to some baseline classifier.

5.1 Datasets

All data sets used in this experimental setup are indefinite with different spectral properties. If the data are given as dissimilarities, a corresponding similarity matrix can be obtained by double centering, as mentioned before [12]. The datasets used for the experiments are described in the following, with details given in the references.

1. **Balls3d/50d** has 200/2000 samples in 2/4 classes. Dissimilarities are generated between balls with the shortest distance on the surfaces [12].
2. The Copenhagen **Chromosomes** data consist of 4,200 human chromosomes from 21 classes represented by grey-valued images. These are transferred to strings measuring their silhouettes and compared using an edit distance [10].
3. The **Delft gestures** (1500 points, 20 classes, signature: (963,536,1)), taken from [1], is a set of dissimilarities generated from a sign-language interpretation problem. The dissimilarities are computed by dynamic time-warping.

4. The **Flowcyto** dataset is based on 612 FL3-A DNA flowcytometer histograms from breast cancer tissues labeled in 3 classes. Dissimilarities are computed between normalized histograms using the L1 norm [1].
5. **Protein**: the Protein data set consists of sequence-alignment similarities for 213 proteins and is used for comparing and classifying protein sequences according to its four classes of globins. The signature is (170,40,3), where class one through four contains 72, 72, 39, and 30 points, respectively [4].
6. **Sonatas** dataset consists of 1068 sonatas from 5 composers (classes) encoded as MIDI data and transformed by normalized compression distance [8].
7. **Zongker** dataset is a digit dissimilarity dataset. The dissimilarity measure was computed between 2000 digits in 10 classes, 200 entries each [6].

5.2 Results

We evaluate the performance of the proposed embedding using cG(M)LVQ from Sect. 4, with the fixed length complex-valued embedded data as inputs. The cGLVQ was parametrized once with and once without relevance learning. To show that our approach performs at least as well as classical embedding, the methods were tested equally on data sets with classical embedding following the same Nyström procedure as the complex embedding. We used one prototype per class for the cG(M)LVQ. In the embedding step of Algorithm 1 we set the meta parameter m (# of landmarks) by a rule of thumb: if $N < 1000$, $m = 40$, if $1000 < N < 5000$, $m = 70$, otherwise $m = 100$. As a baseline, we used a nearest neighbor classifier (NN), which is valid for generic, uncorrected input data and embedded data, but very costly due to the storage of the full input matrix. This is particular unattractive if the dataset is large. Experiments were run in a ten-fold cross-validation. Mean prediction accuracy on the holdout test data and the respective standard deviation is reported in Table 1.

If the data were left uncorrected, we often obtained a rather poor result using the nearest neighbor classifier, sometimes even significantly worse compared to (c)GLVQ and (c)GMLVQ (see balls3d, protein, zongker). In some cases, NN had equal or slightly better performance than the two (c)G(M)LVQ variants (Chromosomes, Sonatas). This is due to the spectrum of eigenvalues: Chromosomes has many eigenvalues, which are almost negligible and close to zero. Sonatas has only a few negative eigenvalues and these eigenvalues are also close to zero. Relevance learning (cG**M**LVQ) significantly improves the results compared to cGLVQ without relevance learning. However, even the mere use of the cGLVQ without relevance learning leads to a significant increase in performance compared to the NN with uncorrected data. Therefore, we assume that our embedding approach, is indeed useful since the use of uncorrected non-psd data shows a clear drop in accuracy using NN and the vectorial embedding permits a more flexible weighting of input contributions. In summary, the presented approach, applying an embedding of the indefinite input data into a complex-valued vector space, shows promising results on a variety of data sets.

Table 1. Prediction accuracy (mean ± standard-deviation) for the GLVQ/cGLVQ variants and the nearest neighbor classifier.

	Without embedding	Classic (real) embedding			Proposed (complex) embedding		
Dataset	NN	NN	GLVQ	GMLVQ	NN	cGLVQ	cGMLVQ
Balls3d	0.49 ±0.06	0.54 ±0.13	0.78 ±0.08	0.98 ±0.02	0.54 ±0.13	0.67 ±0.12	**1.0 ±0.0**
Balls50d	0.25 ±0.04	0.26 ±0.04	0.52 ±0.04	**0.78 ±0.18**	0.26 ±0.04	0.28 ±0.03	0.54 ±0.11
Chromosomes	**0.95 ±0.01**	0.92 ±0.02	0.91 ±0.01	0.94 ±0.02	0.92 ±0.02	0.92 ±0.01	0.94 ±0.01
DelftGestures	**0.96 ±0.02**	0.88 ±0.01	0.94 ±0.01	0.96 ±0.02	0.87 ±0.04	0.95 ±0.01	**0.96 ±0.02**
Flowcyto	0.62 ±0.08	0.63 ±0.06	0.62 ±0.06	0.67 ±0.07	0.59 ±0.04	0.66 ±0.07	**0.70 ±0.05**
Protein	0.23 ±0.1	**0.98 ±0.02**	0.93 ±0.05	0.97 ±0.05	**0.98 ±0.02**	0.92 ±0.07	**0.98 ±0.02**
Sonatas	0.89 ±0.02	0.87 ±0.02	0.82 ±0.03	0.88 ±0.03	0.87 ±0.02	0.80 ±0.03	**0.90 ±0.02**
Zongker	0.58 ±0.05	0.68 ±0.09	0.88 ±0.02	0.92 ±0.02	0.70 ±0.06	0.89 ±0.02	**0.93 ±0.02**

6 Conclusions

In this work, we proposed an efficient, complex-valued embedding and a processing pipeline to analyze non-metric or non-psd proximity data. The approach shows very promising performance on a variety of datasets and is easy to employ. A careful combination of approximation techniques, derived by the authors in former work, permits a valid and still effective calculation of the embedding matrix. By processing the embedding matrix, a straight forward, non-modifying out of sample extension is obtained, not available otherwise. The low-rank embedding is fast and has the benefit that the reconstructed matrix approximates the original indefinite kernel with low error; hence all major information in the original data is preserved. In particular we can omit additional modifiers or eigenvalue corrections which are costly and substantially alter the data. The model of the proposed complex embedding implicitly exists in the Krein space. Using learning algorithms for complex-valued data, predictive models can be obtained with low computational costs. In this initial work, we focused on complex-valued G(M)LVQ and Nearest Neighbor to calculate classification models, but this will be extended to other models in future work. Our initial findings show that the suggested complex-valued embedding of indefinite proximity data, combined with complex-valued classifier models, is a very effective

and promising approach favorable over classical alternatives. Therefore, experimental comparisons to more classical eigenspectrum approaches (like clipping, flipping or shifting negative eigenvalues) or to models working in Krein space [11] are also interesting for further research.

References

1. Duin, R.P.: PRTools (2012). http://www.prtools.org
2. Gisbrecht, A., Schleif, F.: Metric and non-metric proximity transformations at linear costs. Neurocomputing **167**, 643–657 (2015)
3. Goldfarb, L.: A unified approach to pattern recognition. Pattern Recogn. **17**(5), 575–582 (1984)
4. Hofmann, T., Buhmann, J.M.: Pairwise data clustering by deterministic annealing. IEEE Trans. Pattern Anal. Mach. Intell. **19**(1), 1–14 (1997)
5. Iosifidis, A., Gabbouj, M.: Nyström-based approximate kernel subspace learning. Pattern Recogn. **57**, 190–197 (2016)
6. Jain, A., Zongker, D.: Representation and recognition of handwritten digits using deformable templates. IEEE Trans. Pattern Anal. Mach. Intell. **19**(12), 1386–1391 (1997)
7. Kar, P., Jain, P.: Similarity-based learning via data driven embeddings. In: Proceedings of Advances in Neural Information Processing Systems 24: 25th NIPS 2011, Granada, Spain, pp. 1998–2006 (2011)
8. Mokbel, B.: Dissimilarity-based learning for complex data. Ph.D. thesis, Bielefeld University (2016). https://nbn-resolving.de/urn:nbn:de:hbz:361-29004254
9. Münch, M., Raab, C., Biehl, M., Schleif, F.: Structure preserving encoding of non-euclidean similarity data. In: Proceedings of 9th ICPRAM 2020, pp. 43–51 (2020)
10. Neuhaus, M., Bunke, H.: Edit distance based kernel functions for structural pattern classification. Pattern Recogn. **39**(10), 1852–1863 (2006)
11. Oglic, D., Gärtner, T.: Scalable learning in reproducing Kernel Krein spaces. In: Proceedings of the 36th ICML 2019, USA, pp. 4912–4921 (2019)
12. Pekalska, E., Duin, R.: The Dissimilarity Representation for Pattern Recognition. World Scientific (2005)
13. Sato, A., Yamada, K.: Generalized Learning Vector Quantization. In: Proceedings of 8th NIPS 1995 (NIPS'95), pp. 423–429. MIT Press, Cambridge, MA, USA (1995)
14. Schleif, F., Tiňo, P.: Indefinite proximity learning: a review. Neural Comput. **27**(10), 2039–2096 (2015)
15. Schneider, P., Biehl, M., Hammer, B.: Adaptive relevance matrices in learning vector quantization. Neural Comput. **21**(12), 3532–3561 (2009)
16. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis and Discovery. Cambridge University Press, Cambridge (2004)
17. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
18. Straat, M., et al.: Learning vector quantization and relevances in complex coefficient space. Neural Comput. Appl. **32**, 18085–18099 (2019)
19. Trabelsi, C., et al.: Deep complex networks. In: 6th ICLR 2018 (2018)
20. van Veen, R., et al.: An application of generalized matrix learning vector quantization in neuroimaging. Comp. Meth. Progr. Biomed. **197**, 105708 (2020)
21. Wirtinger, W.: Zur formalen Theorie der Funktionen von mehr komplexen Veränderlichen. Math. Ann. **97**, 357–376 (1927)
22. Zhang, L., Zhou, W., Jiao, L.: Complex-valued support vector classifiers. Digit. Signal Process. **20**(3), 944–955 (2010)