This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

Kernel-Based Generalized Median Computation for Consensus Learning

Andreas Nienkötter, Xiaoyi Jiang, *Senior Member, IEEE*

Abstract—Computing a consensus object from a set of given objects is a core problem in machine learning and pattern recognition. One popular approach is to formulate it as an optimization problem using the generalized median. Previous methods like the Prototype and Distance-Preserving Embedding methods transform objects into a vector space, solve the generalized median problem in this space, and inversely transform back into the original space. Both of these methods have been successfully applied to a wide range of object domains, where the generalized median problem has inherent high computational complexity (typically $\mathcal{NP}$-hard) and therefore approximate solutions are required. Previously, explicit embedding methods were used in the computation, which often do not reflect the spatial relationship between objects exactly. In this work we introduce a kernel-based generalized median framework that is applicable to both positive definite and indefinite kernels. This framework computes the relationship between objects and its generalized median in kernel space, without the need of an explicit embedding. We show that the spatial relationship between objects is more accurately represented in kernel space than in an explicit vector space using easy-to-compute kernels, and demonstrate superior performance of generalized median computation on datasets of three different domains. A software toolbox resulting from our work is made publicly available to encourage other researchers to explore the generalized median computation and applications.

Index Terms—Consensus learning, generalized median, kernel functions, distance-preserving embedding, vector spaces.

1 Introduction

One commonly used approach to consensus learning is to formulate it as an optimization problem in terms of generalized median computation [1]. Given a set of objects $O = \{o_1, ..., o_n\}$ in domain $\mathcal{O}$ with a distance function $\delta(o_i, o_j)$, the *generalized median* can be expressed as

$$\bar{o} = \arg\min_{o \in \mathcal{O}} \underbrace{\sum_{o_i \in O} \delta(o_i, o)}_{SOD(o)} \tag{1}$$

In other words, the generalized median is an object that has the smallest sum of distances (SOD, also called *consensus error* [2]) to all objects in the input set. Note that the median object is not necessarily part of set O. The generalized median is the formalization of the intuitive averaging. As a simple example, when dealing with real numbers, it corresponds to well-known concepts from statistics. In case of $\delta(p, q) = (p-q)^2,\ p, q \in \mathbb{R}$, the generalized median is simply the arithmetic average of the given numbers. Changing to another distance function $\delta(p, q) = |p - q|$ results in the usual median of numbers.

The generalized median is a general concept and has been studied for numerous problem domains related to a broad range of applications. Examples include rankings [3], phase or orientation data [4], [5], 3D rigid structures [6], 3D rotations [7], [8], clusterings [9], [10], [11], point-sets [12], shapes [13], 3D surfaces [14], image segmentation [15], sequence data (strings) [16], [17], graphs [18], [19], altas construction [20], and Grassmann average [21]. It is often known under different names in applications, where a specific distance is common, for example geometric median in case of Euclidean vector spaces with the Euclidean distance [22], Steiner string for median strings [2], Karcher mean for positive definite matrices [23] or Kemeny consensus for rankings [24].

Although being simple in its definition, the optimization task (1) turns out to be very complex in many domains. Since the median object is not necessary part of the original set, one has to construct a new object from the whole domain space that minimizes the SOD in equation (1). Even for the simple string edit distance, it has been proven to be $\mathcal{NP}$-hard [25]. The same applies to rankings [3], ensemble clustering [26], median graphs [22] and signed permutations [27] using common distance functions, just to name a few examples. Also for the seemingly simple case of $\mathbb{R}^d$ with the Euclidean distance there is no known polynomial-time algorithm, and it is not even known if this problem is in $\mathcal{NP}$ [28].

Given the high computational complexity, approximate solutions are required to calculate the generalized median in reasonable time. The perturbation strategy in [10] for ensemble clustering, for instance, starts with an initial clustering and iteratively moves a single object to a different (possibly empty) cluster until the optimization SOD function associated with the newly created clustering increases. More sophisticated methods include metaheuristic strategies, e.g. genetic algorithms [18] that optimize a set of candidate solutions by combining and mutating them, simulated annealing [29] that optimizes a single candidate by probabilistic optimization, or block coordinate descent [30] minimizing a matrix or vector representation of the generalized median approximation. These methods are typically heavily tailored to a specific domain $\mathcal{O}$ and use intricate domain knowledge in their computation.

The focus of our work lies in domain-independent meth-

• *A. Nienkötter and X. Jiang are with the Faculty of Mathematics and Computer Science, University of Münster, Münster, Germany.*

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

ods that can compute generalized median solutions for any space. There exist only very few such frameworks. The framework for generalized median approximation in [31] is motivated by a lower bound [18] for generalized median computation in metric spaces. Recently, the prototype-embedding approach [22] has received considerable attention to successfully solve a number of $\mathcal{NP}$-hard consensus learning problem instances (strings, graphs, clusterings, biclusterings) with superior quality [1], [16], [22], [32], [33]. In this framework the objects are first embedded into a vector space, where the median computation is much easier than in the general case. The median vector is then transformed back into the original problem space (reconstruction), resulting in an approximation of the generalized median. Further improvement of the prototype-embedding approach includes alternative object reconstruction methods [34] and more accurate distance-preserving embedding methods compared to prototype embedding [1].

This approach can also be applied to structured prediction, which is a broad research topic and has been studied in many domains. In [35], for instance, a tree structure of a house plan is predicted from an advertising text by first labelling the text, computing the intra-label dependencies and finally constructing a tree from these dependencies. Other examples include label ranking prediction [36], human activity recognition [37], and person re-identification [38]. However, these methods and their embeddings (2D dependency edges in the example) are typically highly specialized to the problem at hand. In contrast, the distance embedding approach is a general method applicable to any domain a distance is known.

The popular prototype-embedding approach [1], [22] is based on *explicit* transformation for the embedding purpose. In this work we propose a novel kernel approach to embedding-based generalized median computation by using an *implicit* transformation in terms of kernel functions. We show that it is possible to handle the generalized median computation *without* knowing the dimension of the embedded space and the concrete embedding. We evaluate our method on artificial and real datasets of three different problem domains (strings, clusterings, rankings). Several kernel functions will be studied, in particular some of them have the nice property of provably preserving the pairwise distances after embedding in the implicit vector space, which is a highly desired property for embedding-based generalized median computation [1]. Using these easy-to-compute kernel functions we demonstrate superior performance in terms of the computed median quality compared to the prototype-based explicit embedding method.

The remainder of the paper is structured as follows. In Section 2, we give a statistical interpretation of generalized median (1) as a maximum-likelihood estimator. The prototype-based embedding framework is summarized in Section 3. Its inherent drawbacks are discussed to motivate our current work. Section 4 presents our kernel-based approach to generalized median computation. The experimental evaluation is given in Section 5. A related software toolbox for public use will be described in Section 6. Finally, some discussions in Section 7 conclude the paper.

2 Generalized Median as Maximum-Likelihood Estimator

We start with the simple case of 1D signal. If the true, but unknown, signal value is $\bar{x} \in \mathbb{R}$, the measured signal $x \in \mathbb{R}$ is disturbed by an error $x - \bar{x}$. Then, the probability of a measurement with this error can be expressed by an exponential function of the error. Assuming the probability is quadratic in the error, this leads to the common Gaussian distribution

$$\mathcal{N}(x|\bar{x},\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right)$$

Assuming the error is not quadratic, one can express the probability as the so-called Laplace distribution

$$\mathcal{L}(x|\bar{x},\sigma) = \frac{1}{2\sigma} \exp\left(-\frac{|x-\bar{x}|}{\sigma}\right)$$

These probability distributions are also known as the second and first law of error, and were first proposed by Laplace around 1775 [39]. Although the Gaussian distribution is the more popular one, it has been shown that it does not always reflect the error distribution of natural data, especially if outliers are present [39], [40]. Naturally, the Laplace distribution appears for example in the error of navigational data [41] or financial data [40].

Now we consider an arbitrary space $\mathcal{O}$. Assuming the true, but unknown, object $\bar{o}$, we make observations o and model the error by the distance $\delta(o,\bar{o})$ in analogy of $|x-\bar{x}|$ in the Laplace distribution for $\mathbb{R}$. Then, one can compute the probability for the occurrence of any object o by

$$\mathcal{L}(o|\bar{o},\sigma) = \frac{1}{2\sigma} \exp\left(-\frac{\delta(o,\bar{o})}{\sigma}\right)$$

Using this distribution, one can estimate the optimal object $\bar{o}$ and the distribution factor σ for a set of objects $O = \{o_1, ..., o_n\}$ by maximizing the likelihood function

$$\begin{aligned} L(\bar{o},\sigma|O) &= \prod_{i=1}^{n} \frac{1}{2\sigma} \exp\left(-\frac{\delta(o_i,\bar{o})}{\sigma}\right) \\ &= \left(\frac{1}{2\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}\delta(o_i,\bar{o})}{\sigma}\right) \end{aligned}$$

under the assumption that o_i are independent and identically distributed. This is equivalent to maximizing the logarithm of the likelihood function

$$\log(L(\bar{o},\sigma|O)) = n\log\left(\frac{1}{2\sigma}\right) - \frac{\sum_{i=1}^{n}\delta(o_i,\bar{o})}{\sigma}$$

Since σ and n are constant, an optimal estimation $\bar{o}$ is equal to minimizing the second term (without the negative sign), which corresponds to the definition of generalized median in (1). In a next step the optimal σ can be computed as

$$\sigma = \frac{\sum_{i=1}^{n}\delta(o_i,\bar{o})}{n}$$

by setting $\frac{\partial \log(L(\bar{o},\sigma|O))}{\partial\sigma}$ to be zero, which is the normalized SOD.

Therefore, the generalized median (1) is the optimal estimation in the maximum likelihood sense so that the observed data (objects) O is most probable. Although this elaboration is mathematically simple and known in some

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

specific spaces (e.g. in the case of signed and unsigned permutations the generalized median is the maximum likelihood solution under the Mallows model [24], [42]), adapting it for the general case of arbitrary domains provides us a deeper understanding of the computation and we are not aware of its mention for general spaces in the literature before.

Occasionally, researchers use the definition

$$\bar{o} = \arg\min_{o \in \mathcal{O}} \sum_{o_i \in O} \delta^2(o_i, o)$$

in analogy of $(x - \bar{x})^2$ in the Gaussian distribution for $\mathbb{R}$ [43]. This can be uniformly treated using the definition (1) by defining an auxiliary distance function $\delta^*() = \delta^2()$.

3 Framework of Embedding-Based Generalized Median Computation

The embedding framework [1], [22] for computing the generalized median (1) uses an explicit embedding function $\phi(o_i)$ to transform the objects o_i into vectors in $\mathbb{R}^d$, leading to the problem

$$\bar{x} = \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} \|\phi(o_i) - x\|_2$$

in vector space using the Euclidean distance. This is solved by using the popular Weiszfeld algorithm [44]. This iterative algorithm delivers a good-quality geometric median in reasonable time. Then, an inverse transformation is used to find an object $\bar{o}$ with $\bar{o} \approx \phi^{-1}(\bar{x})$ that can be seen as the inverse of ϕ, therefore solving

$$\bar{o} \approx \phi^{-1}(\bar{x}) = \arg\min_{o \in \mathcal{O}} \|\phi(o) - \bar{x}\|_2$$

This is a special instance of the so-called pre-image problem in kernel-based machine learning [45] which deals with the difficult reverse problem of finding an object in the original space corresponding to a given embedding.

In the following subsections we will first describe the embedding and median computation part of this framework resulting in $\bar{x}$, followed by an extensive review of the reconstruction step realizing the inverse transformation $\phi^{-1}(\bar{x})$.

3.1 Embedding and computation of generalized median in vector space

Prototype-based embedding was initially proposed in [22]. Choosing d prototype objects $p_1, ..., p_d$ from the input set O, this method uses the embedding function

$$\phi(o_i) = (\delta(o_i, p_1),\ \delta(o_i, p_2),\ ...,\ \delta(o_i, p_d))$$

to compute vectors for each object.

Later, it was shown that although easy to compute, the prototype embedding has several drawbacks [1], in particular the lacking preservation of distances in vector space. In that work, these drawbacks were remedied by using better distance-preserving embedding methods. Multidimensional scaling or curvilinear component analysis, for example, actively improve the preservation of pairwise distances between objects, thus leading to less approximation in the later steps and a better quality approximation of the median overall. Although still only an approximation of the true distances, it was shown that using these and other embedding methods greatly improves the median quality through their accurate representation of the spatial structure of objects in vector space.

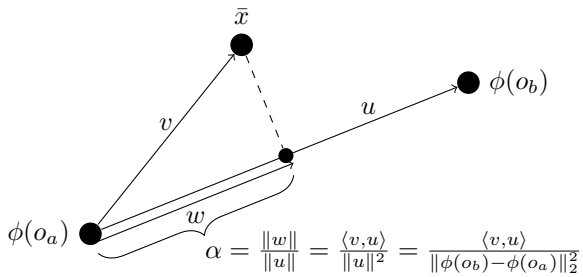


Figure 1. Computation of the ratio α between objects for reconstruction.

Using the computed $\phi(o_i)$ of each object, the generalized median $\bar{x}$ in vector space is then computed using the popular iterative Weiszfeld algorithm [44]

$$\bar{x}_{j+1} = \frac{\sum_{i=1}^{n} \omega_i^j \phi(o_i)}{\sum_{i=1}^{n} \omega_i^j}$$

with weights

$$\omega_i^j = \frac{1}{\|\bar{x}_j - \phi(o_i)\|_2}$$

By selecting the starting vector $\bar{x}_0 \in \mathbb{R}^d$ in specific ways as shown in [46], the Weiszfeld algorithm is proven to converge at a sublinear rate. In practice, however, it is often sufficient to use the mean of $\phi(o_i)$ as easy-to-compute starting point since the algorithm quickly converges to a good approximation of $\bar{x}$ for all but a few trivial-to-resolve starting vectors [47].

3.2 Reconstruction of generalized median from vector space

The inverse function $\phi^{-1}(\bar{x})$ is realized by a reconstruction process involving the nearest neighbors of $\bar{x}$ in the vector space. The simplest reconstruction method – called linear reconstruction – chooses the two nearest neighbors of the median vector, and combines them in a similar fashion to linear interpolation, as seen in Figure 1. Let $\phi(o_a), \phi(o_b)$ be the nearest neighbors of median vector $\bar{x}$, $u = \phi(o_b) - \phi(o_a)$ and $v = \bar{x} - \phi(o_a)$. Then, the reconstruction is achieved by projecting vector v onto u, resulting in a vector $w = \frac{\langle v,u \rangle}{\|u\|}$. The resulting ratio

$$\alpha = \frac{\|w\|}{\|u\|} = \frac{\langle v, u \rangle}{\|\phi(o_b) - \phi(o_a)\|_2^2}$$

between w and u is then used to compute an approximate median object o_m in the original space as an interpolation of o_a, o_b with ratio α. This is achieved with a so-called weighted mean function that constructs a o_m using the following properties

$$\begin{aligned} \delta(o_a, o_m) &= \alpha \cdot \delta(o_a, o_b) \\ \delta(o_m, o_b) &= (1 - \alpha) \cdot \delta(o_a, o_b) \end{aligned} \tag{2}$$

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

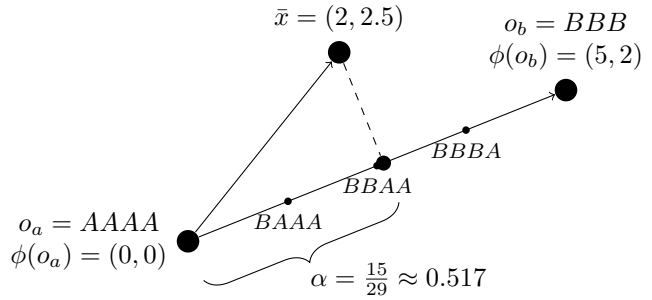


Figure 2. Example for the weighted mean computation using strings. The median object $\hat{o}$ is reconstructed from $\bar{x}$ using the two neighbors $o_a = AAAA$ and $o_b = BBB$ and their respective embeddings. For visualization, possible reconstructed objects using a weighted mean function are shown as points on the line between $\phi(o_a)$ and $\phi(o_b)$.

Under the assumption that the distances were reasonably well preserved in vector space, this approximation should have a lower SOD than the individual objects, similar to the case in vector space. In practice, this weighted mean can often be derived from the distance function $\delta()$ [48], [49], [50].

An example using strings and the common Levenshtein edit distance can be seen in Figure 2. Given a string set O, an embedding $\phi(o)$ was computed that assigns each string a vector in a two-dimensional space. In this case, object $o_a = AAAA$ was assigned $\phi(o_a) = (0, 0)$ and object $o_b = BBB$ was assigned $\phi(o_b) = (5, 2)$. Similarly, all other objects in the set were assigned embedded vectors. After computation of the generalized median in vector space $\bar{x} = (2, 2.5)$, $\phi(o_a)$ and $\phi(o_b)$ were identified as nearest objects. Then, the linear reconstruction method projects $\bar{x}$ onto the line between $\phi(o_a)$ and $\phi(o_b)$ to compute a ratio $\alpha \approx 0.517$. The unknown pre-image $\phi^{-1}(\bar{x})$ is then approximated by constructing a new object o_m that approximates the properties shown in Eq. (2). In the case of the Levenshtein edit distance, a list of minimal necessary edit operations can be obtained to transform o_a into o_b, in this case for example $AAAA \rightarrow BAAA \rightarrow BBAA \rightarrow BBBA \rightarrow BBB$ with corresponding edit distance 4 assuming that all editing steps have cost 1. Therefore, $BBAA$ is selected as approximation of $\phi^{-1}(\bar{x})$ since it is the closest of the intermediate edit steps to fulfilling Eq. (2) with $\delta(o_a, o_m) = 2 \approx 2.06 = \alpha \cdot \delta(o_a, o_b)$ and $\delta(o_m, o_b) = 2 \approx 1.93 = (1 - \alpha) \cdot \delta(o_a, o_b)$. This can be understood as individual editing steps lying on the line between $\phi(o_a)$ and $\phi(o_b)$ and choosing the closest one to the projected $\bar{x}$.

This type of reconstruction relies heavily on the fact that in vector space, the projection of a vector $\bar{x}$ onto the line between $\phi(o_a)$ and $\phi(o_b)$ is the closest point on this line to $\bar{x}$ (see Figures 1 and 2). In embedding based frameworks this is assumed to be true in object space as well and approximated by the weighted mean. Just as in vector space where any point on a line can be expressed as a weighted mean between its end points, a "straight line" in object space can be represented by all objects fulfilling the weighted mean Equation (2). However, depending on the distance formulation, this "line" may not be unique. Using the example above, a second "straight line" of editing steps is $AAAA \rightarrow AAAB \rightarrow AAB \rightarrow BAB \rightarrow BBB$, leading to the weighted mean AAB. This can even happen in vector space using certain distances, for example the Manhattan metric, where an infinite number of possible paths of weighted means exist between two vectors.

Using this definition of "straight line", the orthogonal projection of an object onto the line between two objects is the (not necessarily unique) object which is a weighted mean between o_a and o_b and has the smallest distance to the projected object. However, for many popular distance formulations computing this projection would require to compute a large number of projection candidates that could be very costly itself. For more efficiency of these methods, often only a small number of candidates are computed and the best one is returned. In our case, we will balance the computation of the weighted mean between speed and robustness by only computing two weighted means. The weighted mean from o_a to o_b using α and the weighted mean from o_b to o_a using $1 - \alpha$ are computed, choosing the weighted mean with the better approximation of the median as the result. This is done for both the explicit generalized median computation shown in this section and our kernel based method shown later for a fair comparison.

In addition to the simple linear reconstruction, there are other reconstruction methods, named triangular and recursive reconstruction. The basic idea of these methods is the same, but using three objects instead of two nearest neighbors for triangular reconstruction, or using a more sophisticated projection technique for recursive reconstruction (see [1], [22] for details).

Further improvement of the inverse transformation can be found in [34]. Two new reconstruction methods were proposed, namely linear-recursive and triangular-recursive reconstruction, as well as a method that iteratively improves results of any reconstruction method. Both linear-recursive and triangular-recursive reconstruction still use the same weighted mean function and projection technique, but combine objects in a different way. It could be shown that these methods significantly improve the quality of the reconstructed median over all three previous variants by enhancing the inverse transformation $\phi^{-1}(\bar{x})$.

In summary, the framework of embedding-based generalized median computation comprises three steps: applying an explicit method $\phi(o_i)$ to embed objects from an arbitrary domain into a vector space, computing the median $\bar{x}$ there, and then using an inverse transformation $\phi^{-1}(\bar{x})$ to compute an approximate median object in the original space that corresponds to this median vector.

Despite the popularity and the recent advances, the prototype-embedding approach [1], [22] has some inherent drawbacks due to its nature of *explicit* transformation for the embedding purpose. There is a need of specifying the embedding dimension (number of prototypes) and selecting this number of suitable prototypes. Both may influence the quality of the computed generalized median, but are not always trivial to set. Therefore, we aim to explore instead an approach of *implicit* transformation by means of kernel functions. This approach helps overcome the inherent drawbacks of explicit transformation. More importantly, some of the studied kernel functions have the nice property of provably preserving the pairwise distances after embedding in the implicit vector space, which is fundamental for superior

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

performance of generalized median computation within the embedding-based computation framework [1].

4 Kernel-Based Generalized Median Computation

In this section we present a kernel approach to embedding-based generalized median computation. Despite the *implicit* nature of kernel embedding functions it turns out to be possible to determine all required ingredients for this computation. In particular, we will show that the ratio α that is necessary for the reconstruction combination of objects in $\phi^{-1}()$ can be computed without using an explicit embedding $\phi()$, a kernel function suffices instead. Using the kernel approach, one thus can accurately reconstruct a generalized median without the need of an explicit vector space.

After a brief introduction to kernel methods in Section 4.1, we shortly repeat the main steps of the generalized median computation using explicit transformation in Section 4.2. The main contribution of this work is then described in Section 4.3 (positive definite kernels) and Section 4.4 (indefinite kernels) to perform generalized median computation using implicit transformation in the context of kernel functions. Then, we present several kernel functions in Section 4.5 that will be studied in our experimental work. An analysis of the complexity of the kernel-based median computation framework is given in Section 4.6. Finally, we provide some further discussion in Section 4.7.

4.1 Kernel methods

Kernel functions are well-known from their application in kernel machine [51], support vector machine (SVM), clustering, principal component analysis, etc. [52]. Originally, a kernel function $K(x, y)$ is defined by

$$K\colon\ \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

with a related transformation $\phi : \mathbb{R}^d \to \mathcal{H}_K$ into a Hilbert space $\mathcal{H}_K$ so that $K(x, y) = \langle \phi(x), \phi(y) \rangle$, that is, the kernel function corresponds to the scalar product in $\mathcal{H}_K$. One simple example for the two-dimensional vector space is the kernel function

$$K\colon\ \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R},\ K(x, y) = (xy^\top)^2$$

which computes the scalar product

$$\langle \phi(x), \phi(y) \rangle = \Big\langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (y_1^2, \sqrt{2}y_1y_2, y_2^2) \Big\rangle$$

of a transformation in a three-dimensional vector space. In general, using a kernel function $K(x, y)$, one can compute the scalar product in the transformed vector space without the need of explicitly transforming x and y into $\phi(x)$ and $\phi(y)$, respectively. Since the linear SVM classifier depends on scalar products only, this *kernel trick* allows nonlinear classification of the original vectors by a linear classification in the vector space implied by a kernel function.

The concept of kernel functions for vector space can be extended to an arbitrary space $\mathcal{O}$ to enable classification on other types of data (e.g. strings, trees, graphs) [53], [54], [55], [56]. In this general case, a kernel function is defined by

$$K\colon\ \mathcal{O} \times \mathcal{O} \to \mathbb{R},\ K(o_a, o_b) = \langle \phi(o_a), \phi(o_b) \rangle$$

with a transformation $\phi\colon \mathcal{O} \to \mathcal{H}_K$ that computes the scalar product of the transformed vectors of two objects o_a and o_b. Note that many of these kernels do not have an explicit representation of a vector space and compute the scalar product directly instead.

Since the scalar product in a Hilbert space induces a norm, the distance between two transformed vectors $\phi(o_a)$ and $\phi(o_b)$ can be computed using the kernel function only as follows

$$\begin{aligned}\|\phi(o_a) - \phi(o_b)\|_2 &= \left(\langle \phi(o_a) - \phi(o_b), \phi(o_a) - \phi(o_b) \rangle\right)^{\frac{1}{2}} \\ &= \big(\langle \phi(o_a), \phi(o_a) \rangle - 2\,\langle \phi(o_a), \phi(o_b) \rangle \\ &\quad + \langle \phi(o_b), \phi(o_b) \rangle\big)^{\frac{1}{2}} \\ &= \left(K(o_a, o_a) - 2K(o_a, o_b) + K(o_b, o_b)\right)^{\frac{1}{2}}\end{aligned} \tag{3}$$

4.2 Generalized median computation using explicit transformation

To better understand the kernel-based reconstruction method, we will shortly repeat the central elements of the explicit transformation framework as presented in Section 3. In case of explicitly known transformation ϕ as with the previous prototype and distance-preserving embedding methods [1], the Weiszfeld algorithm [44] is applied to iteratively compute the generalized median $\bar{x}$ in vector space by

$$\bar{x}_{j+1} = \frac{\sum_{i=1}^{n} \omega_i^j \phi(o_i)}{\sum_{i=1}^{n} \omega_i^j} \tag{4}$$

with weights

$$\omega_i^j = \frac{1}{\|\bar{x}_j - \phi(o_i)\|_2} \tag{5}$$

using explicit vectors $\phi(o_i)$ for each object o_i in the input set and nearly any starting vector $\bar{x}_0$. Then, a ratio α of the projection of $\bar{x}$ onto the line between its closest objects $\phi(o_a)$ and $\phi(o_b)$ is computed by

$$\alpha = \frac{\langle \bar{x} - \phi(o_a), \phi(o_b) - \phi(o_a) \rangle}{\|\phi(o_b) - \phi(o_a)\|_2^2} \tag{6}$$

The generalized median $\bar{o}$ is finally reconstructed by computing a weighted mean object between o_a and o_b using this ratio as shown in Section 3.2.

4.3 Generalized median computation using implicit transformation: positive definite kernels

The computation scheme above explicitly needs the transformation $\phi(o_i)$ for each object o_i in the input set and the intermediate result $\bar{x}_j$ after each iteration of the Weiszfeld algorithm. When working with positive definite kernel functions along with an implicit transformation ϕ, however, these fundamental ingredients for median computation are not available. In the following we present a solution to this problem so that we can still use the scheme presented in Section 4.2 to compute the generalized median.

Median computation in kernel space: Let $\bar{x}_j$ be the generalized median vector from the Weiszfeld algorithm at iteration

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

j, see (Eq. (4)). Then, the weight ω_i^j can be computed using Eq. (5) and (3) as

$$\omega_i^j = 1/(\langle \bar{x}_j, \bar{x}_j \rangle - 2\langle \bar{x}_j, \phi(o_i) \rangle + \langle \phi(o_i), \phi(o_i) \rangle)^{\frac{1}{2}} \quad (7)$$

Here $\langle \phi(o_i), \phi(o_i) \rangle$ can be computed using the kernel function $K(o_i, o_i)$. On the other hand, since $\bar{x}_j$ is unknown we are unable to evaluate $\langle \bar{x}_j, \bar{x}_j \rangle$ and $\langle \bar{x}_j, \phi(o_i) \rangle$. Nevertheless, these two items can be precisely determined using the iterative nature of the Weiszfeld algorithm by substituting $\bar{x}_j$ with its previous step of Eq. (4) and using the bilinear properties of the scalar product. This results in

$$\begin{aligned}\langle \bar{x}_j, \bar{x}_j \rangle &= \left\langle \frac{\sum_{u=1}^n \omega_u^{j-1}\phi(o_u)}{\sum_{u=1}^n \omega_u^{j-1}}, \frac{\sum_{u=1}^n \omega_u^{j-1}\phi(o_u)}{\sum_{u=1}^n \omega_u^{j-1}} \right\rangle \\ &= \frac{\sum_{u=1}^n \sum_{v=1}^n \omega_u^{j-1}\omega_v^{j-1} K(o_u, o_v)}{\left(\sum_{u=1}^n \omega_u^{j-1}\right)^2} \end{aligned} \quad (8)$$

and

$$\begin{aligned}\langle \bar{x}_j, \phi(o_i) \rangle &= \left\langle \frac{\sum_{u=1}^n \omega_u^{j-1}\phi(o_u)}{\sum_{u=1}^n \omega_u^{j-1}}, \phi(o_i) \right\rangle \\ &= \frac{\sum_{u=1}^n \omega_u^{j-1} K(o_u, o_i)}{\sum_{u=1}^n \omega_u^{j-1}} \end{aligned} \quad (9)$$

Note that the right hand side of Eq. (8) and (9) only contains kernels between input objects and the weights of the Weiszfeld algorithm in the previous step of the iteration. Bringing Eq. (7), (8) and (9) together, one can therefore compute ω_i^j iteratively from kernels between input objects without explicit knowledge of the transformed vectors $\phi(o_i)$ and the immediate median $\bar{x}_j$. As a starting value for this iteration scheme, one can for example set all $\omega_i^0 = 1$. This corresponds to the mean of objects in kernel space and is often a good first approximation of the median result. It is important to emphasize that we are only able to use the Weiszfeld algorithm to compute the weights ω_i^j, but not the immediate median $\bar{x}_j$. As will be shown in the following, this is sufficient for reconstructing the generalized median in the original space.

Reconstruction: Even without explicitly knowing the median $\bar{x}$ in the kernel space, we need to reconstruct the generalized median $\bar{o} = \phi^{-1}(\bar{x})$ in the original space. Our solution starts with the insight that for the reconstruction, we actually do not really need to explicitly know the median $\bar{x}$. It suffices to know the nearest neighbors of $\bar{x}$ and the related ratio α in Eq. (6). Their corresponding input objects are then combined to build the generalized median $\bar{o}$. The nearest neighbors of $\bar{x}$ can be easily found by the final weights ω_i after convergence of the Weiszfeld algorithm since Eq. (5) implies that these weights are the inverse of the norm between object o_i and the median in kernel space. That is, sorting the final weights ω_i in a descending order will deliver the nearest neighbors of the unknown $\bar{x}$.

The final weights ω_i are related to the unknown median $\bar{x}$ in kernel space by

$$\bar{x} = \frac{\sum_{i=1}^n \omega_i \phi(o_i)}{\sum_{i=1}^n \omega_i} \quad (10)$$

Algorithm 1 Kernel-based Linear ($l = 2$) and Triangular ($l = 3$) Reconstruction

Input: Object set O, integer l, final weights ω_i, weighted mean function $wm()$
Output: Median object $\bar{o}$

```
Select o_1, ..., o_l with the l maximal ω_i
ō_1 = o_1
for j = 2 to l do
    Compute α using Eq. (11) with objects ō_{j-1}, o_j
    ō_j = wm(ō_{j-1}, o_j, α)
end for
return ō_l with the related SOD
```

Given two nearest neighbors o_a and o_b, inserting Eq. (10) into Eq. (6) leads to

$$\begin{aligned}\alpha &= \frac{\left\langle \frac{\sum_{i=1}^n \omega_i \phi(o_i)}{\sum_{i=1}^n \omega_i} - \phi(o_a), \phi(o_b) - \phi(o_a) \right\rangle}{||\phi(o_b) - \phi(o_a)||^2} \\ &= \frac{\frac{\sum_{i=1}^n \omega_i (K(o_i, o_b) - K(o_i, o_a))}{\sum_{i=1}^n \omega_i} - K(o_a, o_b) + K(o_a, o_a)}{K(o_b, o_b) - 2K(o_b, o_a) + K(o_a, o_a)} \end{aligned} \quad (11)$$

This computation only needs the final weights w_i and kernel values between all objects in the set as well as o_a and o_b.

The simplest method, linear reconstruction [1], [22], uses the two nearest neighbors o_1 and o_2 for the reconstruction. Starting with the – in kernel space – closest object to the median $\bar{o}_1 = o_1$, the α value between it and the next closest object o_2 is computed and applied in the weighted mean function to generate a better approximation $\bar{o}_2$ of the median object. For the triangular reconstruction, a third nearest neighbor o_3 is needed and another α is computed between the median object $\bar{o}_2$ and o_3 towards a refined median object $\bar{o}_3$. The returned median approximation is the last constructed object $\bar{o}_l$. This is shown in Algorithm 1, using $l = 2$ for linear and $l = 3$ for triangular reconstruction.

Algorithm 2 Kernel-based Linear (Triangular) Recursive Reconstruction

Input: Object set O, final weights ω_i, weighted mean function $wm()$
Output: Median object $\bar{o}$

```
ō_best = ∅
while |O| > 1 do
    Divide O into |O|/2 pairs (|O|/3 triples) by grouping
    maximal ω_i first.
    O' = ∅
    for each pair (o_a, o_b) (triple (o_a, o_b, o_c)) do
        Compute ō using Algorithm 1 with l = 2 (l = 3)
        O' = O' ∪ {ō}
    end for
    ō_best = arg min_{o ∈ {ō_best} ∪ O'} SOD(o)
    O = O'
end while
return ō_best with the related SOD
```

We were not able to adapt the recursive and best-recursive reconstruction methods [1], [22] for kernel methods due to the recursive projection onto hyperplanes. There-

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

fore, we leave these methods out and instead adapted linear recursive and triangular recursive reconstruction described in [34]. For prototype embedding, they show superior results compared to best-recursive reconstruction anyway and therefore should have a similar performance for kernel methods. Both methods are shown in Algorithm 2. First, the object set is divided into pairs (or triples) of objects, and a linear (triangular) reconstruction is performed for each case. This results in a new set of $|O|/2$ ($|O|/3$) objects. The algorithm is repeated using this new set until only one object remains. The object with the lowest SOD of all computed objects is returned as the approximated median.

Overall algorithm: The overview of the proposed kernel-based generalized median framework is shown in Algorithm 3. First in lines 1 to 6, the necessary weights of the objects are determined using one of the kernel methods to be discussed in Section 4.5. Note that in contrast to previous explicit embedding, neither the vectors $\phi(o_i)$ nor the median vector $\bar{x}$ can be explicitly computed. As for the number j_{max} of iterations of the kernel-Weiszfeld algorithm, only a low number of iterations is needed in practice and the computation can be stopped early once ω_i^j converges (see Section 5.3). These weights are then used in one of the reconstruction methods to compute an approximate generalized median in the original space.

Algorithm 3 Kernel-Based Generalized Median Framework

Input: Object set O, distance function $\delta()$, weighted mean function $wm()$, kernel function $K()$
Output: Median object $\bar{o}$

```
   /* Computation of median weights */
1: Initialize ω_i^0 = 1 for all 1 ≤ i ≤ |O|
2: for j = 1 to j_max do
3:    for i = 1 to n do
4:       Compute ω_i^j using Eq. (7), (8) and (9)
5:    end for
6: end for
   /* Reconstruction */
7: Compute ō using reconstruction algorithm 1 or 2
8: return ō with the related SOD
```

4.4 Generalized median computation using implicit transformation: indefinite kernels

In the previous section we have shown the computation of the generalized median in kernel space using positive definite kernel functions. However, a number of kernel functions that can be applied to any domain given a distance function (as shown in Section 4.5 and later used in our evaluation) are not positive definite and therefore do not fulfill some basic assumptions used in the previous section. This is similar to the case of kernel SVM, where these kernels are used for classification even though they are not guaranteed to be positive definite [57]. To ensure the general nature of this framework, we will attempt to approximate the generalized median using these indefinite kernels as well. For this reason we will shortly introduce the concept of pseudo-Euclidean spaces and show that indefinite kernel functions can be used to compute the generalized median in such a pseudo-Euclidean space instead of normal Euclidean space.

Pseudo-Euclidean spaces are linear vector spaces including an indefinite, symmetric bilinear form $\langle \cdot, \cdot \rangle$ and can be expressed as $\mathbb{E} = \mathbb{R}^{(p,q)} = \mathbb{R}^p \times i\mathbb{R}^q$, where i is the imaginary unit, i.e. as spaces whose vectors consist of p real and q imaginary elements [58]. As such, pseudo-Euclidean spaces are subspaces of the complex vector space $\mathbb{C}^{p+q}$. In contrast to standard Euclidean spaces, this allows for negative squared distances $\delta^2 : O \times O \to \mathbb{R}$ with

$$\delta^2(o_i, o_j) = \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle .$$

According to [57], [58], any symmetric indefinite kernel function $K(o_i, o_j)$ can be used to construct such a real squared distance by substituting the inner product with indefinite kernels

$$\delta^2(o_i, o_j) = K(o_i, o_i) - 2K(o_i, o_j) + K(o_j, o_j) \tag{12}$$

Additionally, if this squared distance is symmetric with $\delta^2(o_i, o_i) = 0$ for all $x \in O$, which is true for Equation (12) for any symmetric function K, one can express δ^2 as

$$\delta^2(o_i, o_j) = \|\phi(o_i) - \phi(o_j)\|_2^2 \tag{13}$$

using a transformation $\phi : O \to \mathbb{E}$ into a pseudo-Euclidean space $\mathbb{E}$. As such, indefinite kernels allow the computation of a real-valued squared norm in a pseudo-Euclidean space, which, as the square root of a real number, translates to a complex valued norm consisting only of a real *or* imaginary part.

As we need the norm in the computation of the weights of the Weiszfeld algorithm as shown in Equation (5), we will need to use the root of the above squared distance shown in Equation (13) and as such expand the pseudo-Euclidean space to a full complex space. This leads to modifications in Equations (7), (8) and (9) for the iterative computation of weights using complex norms. In contrast to real spaces, the inner product in complex spaces is conjugate symmetric ($\langle x, y \rangle = \overline{\langle y, x \rangle}$) and conjugate linear in the second argument ($\langle x, \lambda y \rangle = \overline{\lambda} \langle x, y \rangle$).

Using the conjugate symmetry, Equation (7) becomes

$$\begin{aligned} \omega_i^j = & 1/(\langle \bar{x}_j, \bar{x}_j \rangle - \langle \bar{x}_j, \phi(o_i) \rangle \\ & - \overline{\langle \bar{x}_j, \phi(o_i) \rangle} + \langle \phi(o_i), \phi(o_i) \rangle)^{\frac{1}{2}} \end{aligned} \tag{14}$$

while, using conjugate linearity in the second argument, Equation 8 becomes

$$\begin{aligned} \langle \bar{x}_j, \bar{x}_j \rangle &= \left\langle \frac{\sum_{u=1}^n \omega_u^{j-1} \phi(o_u)}{\sum_{u=1}^n \omega_u^{j-1}}, \frac{\sum_{u=1}^n \omega_u^{j-1} \phi(o_u)}{\sum_{u=1}^n \omega_u^{j-1}} \right\rangle \\ &= \frac{\sum_{u=1}^n \sum_{v=1}^n \omega_u^{j-1} \overline{\omega_v^{j-1}} K(o_u, o_v)}{\left(\sum_{u=1}^n \omega_u^{j-1}\right) \left(\overline{\sum_{u=1}^n \omega_u^{j-1}}\right)} \end{aligned} \tag{15}$$

Equation (9) remains unchanged. Note that since the complex conjugate does not affect real numbers, these equations can also be used for positive definite kernels. As these modifications may influence the convergence of the Weiszfeld algorithm, we will experimentally study its convergence using indefinite kernels in Section 5.3.

All in all, this leads to complex weights in the computation of the Weiszfeld algorithm, which – inserted into the

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

also unchanged Equation (11) – can lead to a complex α value. However, for the computation of the weighted mean real value is needed. As such, in the case of a complex α, we will use its magnitude to compute the weighted mean.

4.5 Kernel functions

We briefly describe a variety of kernel functions used in our experimental work, both positive definite domain-dependent kernels and indefinite domain-independent kernels. In addition, we show that several special cases of these kernel functions are able to preserve the distance in the vector space of their respective projection, thus making them ideal candidates for kernel-based median computation.

For a general framework for generalized median computation one would like to use distance-preserving domain-independent positive definite kernel functions to ensure a good distance-preserving embedding in an implicit kernel space where the median is approximated, similar to the argument for distance-preserving explicit embedding methods in [1]. However, we were not able to find appropriate positive definite kernel functions that could be used for every domain. Therefore, we will first present a number of positive definite domain-dependent kernels that can be used in this framework, followed by a number of indefinite kernel functions that can be used in any domain. In the end, we will discuss the distance preservation of these kernels.

4.5.1 Positive definite domain-dependent kernels

Although positive definite domain-independent kernels would be preferable for our method shown in Section 4.3, we were only able to find positive definite kernels for specific domains. Therefore, we will present three domain specific kernel functions in this section.

String subsequence kernel: The string subsequence kernel K^{ssk} [59] is a popular kernel function in string based applications for strings s of alphabet Σ. Given the set $\{u_1, u_2, ...\} \subset \Sigma^{|u|}$ of all possible substrings u_i with fixed length $|u|$, the transformation of a string s is defined as

$$\phi^{ssk}(s) = (\phi_{u_1}(s), \phi_{u_2}(s), ...), \quad \phi_u(s) \quad = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}$$

where $\mathbf{i} = (i_1, ..., i_{|u|})$ are increasing indices, $s[\mathbf{i}]$ the substring of s consisting only of the symbols at positions in $\mathbf{i}$, and $l(\mathbf{i}) = i_{|u|} - i_1 + 1$ is the length of the substring containing $\mathbf{i}$. $\lambda \leq 1$ is a weighting parameter. As such, each value in vector $\phi^{ssk}(s)$ encodes the frequency and compactness of the occurrences of a substring u_i in string s. Interestingly, the kernel $K^{ssk}(s_1, s_2) = \langle \phi^{ssk}(s_1), \phi^{ssk}(s_2) \rangle$ can be computed recursively without the need to explicitly compute the embedding $\phi^{ssk}(s)$ [59]. In our evaluation, we chose $|u| = 2$ and $\lambda = 0.5$.

Partition kernel: The partition kernel is designed for the domain of clusterings and based on a simple transformation

$$\phi^{part}(x) = (\phi^p_{1,2}(x), ..., \phi^p_{1,n}(x), \phi^p_{2,3}(x), ..., \phi^p_{n,n}(x))$$
$$\phi^p_{i,j}(x) = \begin{cases} 1, & x(i) = x(j) \\ 0, & \text{otherwise} \end{cases}$$

where x is a vector containing the clustering labels of n elements. If done for two clusterings, the scalar product $K^{part}(o_a, o_b) = \langle \phi^{part}(o_a), \phi^{part}(o_b) \rangle$ is the number of pairs whose cluster labels are equal in o_a and o_b, while the label itself can be different. As a scalar product between two vector transformations, it is guaranteed to be a positive definite kernel function.

Kendall kernel: The Kendall kernel K^{kend} [60] is designed for the domain of permutations and based on the Kendall-τ distance between permutations. This distance counts the number of pairs whose order is different between two permutations. For two permutations, the Kendall kernel is the difference between the number of elements that have the same order in both permutations and the number of elements that have different orders in both permutations, normalized by the number of pairs.

4.5.2 Indefinite domain-independent kernels

As our method is supposed to be used in a large variety of domains, we will present here a number of kernel methods that can be used with any distance function. However, these kernel functions are indefinite for a wide range of distances, including the ones used in our evaluation.

Distance substitution kernels: Haasdonk and Bahlmann [61] showed that given a distance function δ and an origin object o, one can induce a scalar product between these objects

$$\langle o_a, o_b \rangle^o_\delta = \frac{1}{2} \left(\delta(o_a, o)^2 + \delta(o_b, o)^2 - \delta(o_a, o_b)^2 \right) \qquad (16)$$

Using this scalar product, they proposed four kernel functions in object space

$$K^{lin}_\delta(o_a, o_b) = \langle o_a, o_b \rangle^o_\delta$$
$$K^{nd}_\delta(o_a, o_b) = -\delta(o_a, o_b)^\beta, \qquad \beta \in [0, 2]$$
$$K^{pol}_\delta(o_a, o_b) = (1 + \gamma \langle o_a, o_b \rangle^o_\delta)^p, \qquad \gamma \in \mathbb{R}^+, p \in \mathbb{N}$$
$$K^{rbf}_\delta(o_a, o_b) = e^{-\gamma \delta(o_a, o_b)^2}, \qquad \gamma \in \mathbb{R}^+$$

Note that only weak assumptions are imposed on the distance function δ: non-negative, symmetric, zero diagonal (i.e. $\delta(x, x) = 0$). If some given distance function does not satisfy these requirements, it can easily be transformed to satisfy them. Concretely, it can be symmetrized by $\bar{\delta}(x, y) := \frac{1}{2}(\delta(x, y) + \delta(y, x))$, given zero diagonal by $\bar{\delta}(x, y) := \delta(x, y) - \frac{1}{2}(\delta(x, x) + \delta(y, y))$, and made positive by $\bar{\delta}(x, y) := |\delta(x, y)|$ [61]. Thus, the distance substitution kernels above can be applied in conjunction with any distance function. However, it is only a positive definite kernel (or conditionally positive definite for K^{nd}_δ) if the distance function is negative definite, i.e. isometric to an L_2-norm [61]. As a large number of distance functions, for example the string edit distance, are not isometric to an L_2-norm, these kernels are often indefinite.

Combination kernel: Specifically for the domain of strings using the edit distance, Neuhaus and Bunke [62] proposed a combination of the above scalar product (Eq. (16)) using several origin objects $o_i \in \bar{O} \subset O$

$$K^+_S(o_a, o_b) = \sum_{o_i \in \bar{O}} \langle o_a, o_b \rangle^o_\delta$$
$$K^*_S(o_a, o_b) = \prod_{o_i \in \bar{O}} \langle o_a, o_b \rangle^o_\delta$$

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

Using a subset of strings instead of only one origin object, they aim to reduce the dependence on this reference string. Note that no edit distance specific properties were used, and both kernels can be used with any distance function δ similar to the distance substitution kernels. From both combination methods we propose to use $K_\delta^{comb} = K_S^+$ due to its distance-preserving nature, as shown in section 4.5.3.

4.5.3 Distance-preserving kernels

Several of the above shown kernel functions are able to preserve the distance in the vector space of their respective projection. Although having distance-preserving positive definite kernels would be preferable, the above mentioned kernel functions K^{ssk}, K^{part} and K^{kend} are not. K^{ssk} has no direct relation to the string edit distance at all, while for K^{part} and K^{kend} the distance is proportional to the squared norm in kernel space (a proof for K^{kend} is given in [60]).

In the following we will show how a number of the previous shown indefinite kernels are distance-preserving for specific parameters. Using Eq. (3), one can show that K_δ^{comb} and K_δ^{lin} preserves distances in kernel space

$$\begin{aligned}
\|\phi(o_a) - \phi(o_b)\|_2 &= \Big(K_\delta^{comb}(o_a,o_a) - 2K_\delta^{comb}(o_a,o_b) \\
&\quad + K_\delta^{comb}(o_b,o_b)\Big)^{\frac{1}{2}} \\
&= \Bigg(\sum_{o_i \in \bar{O}} \Bigg(\frac{\delta(o_a,o_i)^2}{2} + \frac{\delta(o_a,o_i)^2}{2} - \frac{\delta(o_a,o_a)^2}{2} \\
&\quad - \delta(o_a,o_i)^2 - \delta(o_b,o_i)^2 + \delta(o_a,o_b)^2 \\
&\quad + \frac{\delta(o_b,o_i)^2}{2} + \frac{\delta(o_b,o_i)^2}{2} - \frac{\delta(o_b,o_b)^2}{2}\Bigg)\Bigg)^{\frac{1}{2}} \\
&= \left(\sum_{o_i \in \bar{O}} \delta(o_a,o_b)^2\right)^{\frac{1}{2}} \\
&= \sqrt{|\bar{O}|}\, \delta(o_a,o_b)
\end{aligned}$$

In the special case of $|\bar{O}| = 1$, this is the K_δ^{lin} kernel. For $\bar{O} > 1$, the distortion is constant which has no influence on the minimization of the median function (1), meaning that it is an optimal distance-preserving embedding in kernel space. Similarly, it can be shown that using K_δ^{nd}, the embedding leads to

$$\|\phi(o_a) - \phi(o_b)\|_2 = \sqrt{2}\delta(o_a,o_b)^{\frac{\beta}{2}}$$

meaning that it is distance-preserving for $\beta = 2$. For K_δ^{pol} using $p = 1$ it leads to

$$\|\phi(o_a) - \phi(o_b)\|_2 = \sqrt{\gamma}\delta(o_a,o_b)$$

and is therefore distance-preserving for $p = 1$ and any $\gamma \neq 0$, in particular $\gamma = 1$.

The observation above makes all four distance-preserving options ideal candidates for kernel-based median computation using the above mentioned parameters, since there are no distortions between distances in kernel space. This should lead to a better median computation in kernel space and ultimately better median reconstruction, which is indeed confirmed by the experimental results reported in Section 5.

Table 1
Complexity of the kernel-based median computation framework

Algorithm steps	Complexity
Precomputing kernel values $K(o_i, o_j)$	$\mathcal{O}(n^2)$
Precomputing weights ω_i	$\mathcal{O}(j_{max} \cdot n^2)$
Linear/triangular reconstruction	$\mathcal{O}(n)$
Linear/triangular recursive reconstruction	$\mathcal{O}(n \log n)$

4.6 Complexity of the proposed framework

The asymptotic complexity of the proposed methods can be seen in Table 1. Assuming the weighted mean function and kernel function can be evaluated in constant time, one needs in total $O(j_{max} \cdot n^2)$ time for precomputing the kernel values and weights ω_i, as these do not change in the reconstruction process. Note that j_{max} is quite low in practice (see the evaluation in Section 5.3). For linear and triangular reconstruction (Algorithm 1) additional $O(n)$ time is needed for evaluating α a constant number of times with $O(n)$ each. Similarly, $\mathcal{O}(n \log n)$ additional time is needed for linear and triangular recursive reconstruction (Algorithm 2) by $O(\log n)$ evaluations of α. Overall, our algorithm thus has the complexity $O(n^2)$.

4.7 Discussions

In the previous sections we have shown how the generalized median can be approximated using positive definite and indefinite kernel functions, and given a number of examples for kernel functions that could be used using any distance function. Although a few of approximations are made, among which are the transfer of α computed by the orthogonal projection in vector space onto an object space where there might not be such a clear projection, or the use of indefinite kernels where the Weiszfeld algorithm may not find a minimum or even converge, or the computation of α using complex weights whose interpretation is not clear. Thus, we will evaluate the convergence of the Weiszfeld algorithm using indefinite kernels in Section 5.3 and the computation of α in kernel space in Section 5.4 to show that our assumptions are reasonable.

In addition, it should be noted that using a kernel to compute the generalized median in kernel space can change the distance at hand. I.e. instead of minimizing the sum of distances, we minimize the sum of distances of embedded objects. Therefore, it is important to use kernel functions related to the distance in the original space.

5 Evaluation

In this section we present the experimental results using the method presented in Section 4. First, we introduce the datasets used in our study in Section 5.1. In Section 5.2 we will show how the previously discussed kernel functions preserve distances in kernel space, and give reasons for why they are expected to produce better results than other methods. Section 5.3 investigates the convergence of the Weiszfeld algorithm using indefinite kernels, while Section 5.4 studies the reconstruction of the median object using the weighted mean. Section 5.5 compares the results of the proposed kernel-based method to the previous Prototype [22]

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

Table 2
Evaluated datasets for generalized median computation

Dataset	Type	#Sets	#Objects in each set	Distance function
Darwin	String	36	40	Levenshtein
CCD	String	22	100	Levenshtein
UCI Cluster	Cluster	8	25	Partition Distance
Gen. Cluster	Cluster	8	20	Partition Distance
ranking-bio	Ranking w. Ties	40	7	gen. Kendall-τ
ranking-real	Ranking w. Ties	40	8-17 (mean 12)	gen. Kendall-τ

and Distance-Preserving Embedding Frameworks [1] with explicit vector space embedding. Then, we study the correlation of the median quality (in terms of sum of distances) and the degree of distance preservation in Section 5.6. Finally, a discussion of the computational time is presented in Section 5.7.

5.1 Datasets

Table 2 shows the datasets used in our study. To ensure a wide range of applications we tested 6 datasets divided into three types, from which two were artificially generated and four real. The exact number of sets in each dataset and number of objects in each set can be found in the table.

The string datasets Darwin and CCD consist of character strings using the Levenshtein edit distance for the median computation. The Darwin dataset is artificially generated using lines of Charles Darwins famous work "On the Origin of Species" that were randomly modified by substitution, insertion and deletion of characters according to real-life OCR error rates [16]. The Copenhagen Chromosome Dataset (CCD) is a dataset containing encoded parts of chromosome sequences [63]. In both cases the corresponding weighted mean function applies string edit operations from the Levenshtein edit distance until the ratio in Eq. (2) is reached [49], as seen in Section 3.2.

The UCI Cluster and Gen. Cluster datasets contain integer label vectors encoding clusterings of unknown data. The UCI Cluster dataset was created by clustering data from the UCI Data Repository [64] with k-means clustering using varying parameters. Each set of the Gen. Cluster dataset consists of a random base label vector that was modified by random changes in labels using the method in [33]. Here, the Partition Distance was used, which counts the number of objects that have to swap their assigned cluster into a different one for both clusterings to become the same partition of the data. The labels of each cluster are disregarded. For the weighted mean function, we change round$(\alpha \cdot \delta(o_a, o_b))$ of disagreeing labels of the first cluster o_a into the corresponding labels of the second cluster o_b [50].

The ranking-bio and ranking-real datasets are two real world datasets [65] and contain ranking with ties. The first one, ranking-bio, consists of the BioMedical dataset, which includes only a very small number of real world rankings in each set. The ranking-real dataset is composed of the F1, Ski Cross, Ski Jumping and WebSearch datasets. These sets consist of the results of different sport tournaments and the order of search results in different web search engines. In both cases the generalized Kendall-τ distance is used [3]. This distance function measures the number of disagreements between rankings. For example, if x is ranked ahead of y in the first ranking, but y is ranked ahead in the second, it is counted as one disagreement, otherwise not. Since ties are a possibility, it is counted as 0.5 disagreements if x and y are tied in one ranking but not tied in the other. The distance is the summed number of disagreements between all ranked items. As weighted mean function, starting from o_a we iteratively move ranked items with the largest number of disagreements into new positions with lower number of disagreements towards o_b, until the required ratio α is reached. Note that there exist simple methods such as the classical Borda and Copeland method for median computation in ranking space. It can be easily shown that these methods are the generalized median computation (1) based on simple distance functions. In our work we use the more sophisticated generalized Kendall-τ distance, which leads to a $\mathcal{NP}$-hard median computation problem [3]. Thus, our approach is particularly suitable to apply.

It should be pointed out that these distance functions are not necessarily negative definite, as is the requirement of the distance substitution kernels to be positive definite. Using the modifications shown in Section 4.4, we can however still apply our method using these kernel functions to compute the generalized median in kernel space. As will be seen in the results, the kernel method is still able to compute a good approximation of the generalized median for such kernels.

5.2 Distance preservation of kernel functions

Since the generalized median computation considerably benefits from a good preservation of distances [1], we measured the degree of distance preservation of the kernels. For each kernel introduced in Section 4.5, we computed all pairwise distances in kernel space using Eq. (3), and determined the ratio c between original distance and Euclidean distance in vector space

$$\delta(o_i, o_j) = c \cdot \|\phi(o_i) - \phi(o_j)\|_2 \tag{17}$$

A perfect distance-preserving embedding is achieved if there is a constant c for all object pairs.

For comparison, we used the embedding method CCA, which showed the best distance preservation for these datasets [1]. The results for the Darwin dataset as an example can be seen in Figure 3. For all object sets and pairwise objects of this dataset these histograms show the distribution of the constant c in Eq. (17). The more c is located at one constant, the better the distances are preserved in the implicit embedding of the kernel or the explicit embedding of CCA. As theoretically shown in Section 4.5.3, K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} achieve a perfect distance-preserving embedding. K_δ^{rbf} however does not guarantee a distance-preserving embedding and is shown to be slightly worse than CCA. Similar observation could also be made on the other five datasets. Thus, we expect that K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} will have the most accurate object representation in vector space and accordingly the best median result.

5.3 Convergence of the Weiszfeld algorithm for indefinite kernels

As shown in Section 4.5, many domain-independent kernels are not necessarily positive definite, leading to complex

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

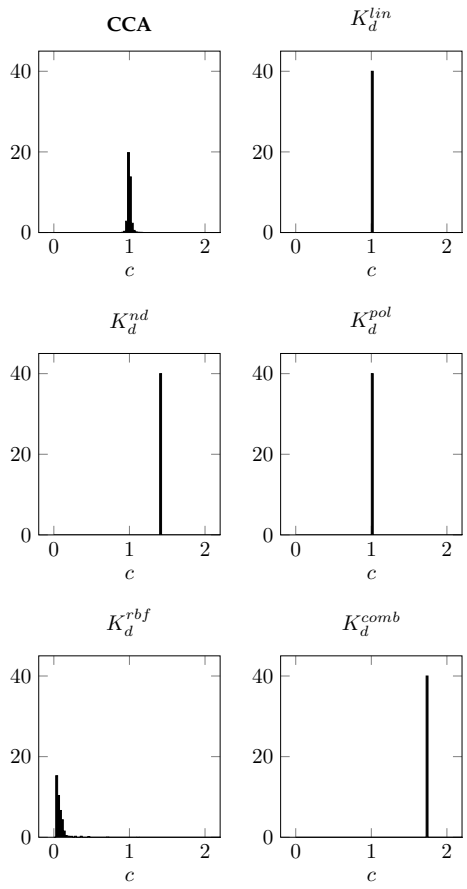


Figure 3. Histogram of the distance distortion constant c for the Darwin dataset. The used parameters for kernel methods are $\beta = 2$ for K_δ^{nd}, $\gamma = 1$ and $p = 1$ for K_δ^{pol}, and $|\bar{O}| = 3$ for K_δ^{comb}. Explicit embedding methods used $0.8 \cdot n$ dimensions where n is the number of objects.

Table 3
Convergence of the Weiszfeld algorithm for 5 indefinite kernel functions K_δ^{lin}, K_δ^{nd}, K_δ^{pol}, K_δ^{rbf}, K_δ^{comb} (first value in each column) and 3 positive definite kernel functions K^{ssk}, K^{part}, K^{kend} (second value in each column).

Dataset (num sets)	max-iter	med-iter	complex weights
Darwin (36)	29 / 13	9 / 11	28/0 (of 180)
CCD (22)	17 / 33	12 / 24	0/0 (of 110)
Gen-Cluster (8)	14 / 12	10 /10	0/0 (of 40)
UCI-Cluster (8)	27 / 19	20 / 15	0/0 (of 40)
ranking-bio (40)	144 / 85	42 / 76	2/0 (of 200)
ranking-real (40)	31 /23	17 / 17	0/0 (of 200)
random-string (1000)	11 / 11	10 / 10	0/0 (of 5000)
random-cluster (1000)	14 / 11	12 / 10	0/0 (of 5000)
random-ranking (1000)	8 / 9	6 / 8	0/0 (of 5000)

values in the computation of the Weiszfeld algorithm if used as is (Section 4.4). For vector spaces – and therefore positive definite kernels – the convergence of the Weiszfeld algorithm is well understood [66]. However, these convergence proofs do not hold in the case of indefinite kernel functions. As we are unable to formally prove the convergence in this case, we conducted a simulation to test if the algorithm converges for a set of given objects using infinite kernels.

In addition to the datasets shown in Section 5.1, we generated 1000 sets of randomly generated strings, clusterings and rankings each for this test. Each randomly generated set consists of 50 to 150 objects of size 50 to 100. Here, the size is the number of characters for strings, the number of clustered objects for clusterings and the number of ranked values for rankings, respectively. For each string in the random-string dataset, random letters were uniformly drawn from all letters of the english alphabet. Each clustering in random-clustering was created by uniformly drawing random integer labels to fill a label vector. The number of clusters varies between 3 and 10. For rankings, a random permutation was created, again drawn uniformly from all possible permutations of a given length.

For each set we computed the weights of the generalized median in kernel space using Equation (7) for all five presented (possibly) indefinite kernels K_δ^{lin}, K_δ^{nd}, K_δ^{pol}, K_δ^{rbf} and K_δ^{comb} as well as three positive definite kernels K^{ssk}, K^{part} and K^{kend} (Section 4.5).

Table 3 shows the resulting statistics for our test. For each dataset this table reports the maximum (median) number of iterations for convergence, max-iter (med-iter), over all tested sets, as well as the absolute amount of complex-valued weights ω_i (Eq. 14) that appeared over all datasets (complex weights). The first value in each column is computed only using indefinite kernels shown in Section 4.5.2, while the second value is computed using the domain-dependent positive definite kernel as described in Section 4.5.1. In all datasets using all kernels, the Weiszfeld algorithm in the form shown in Equation (7) converges in a relatively short amount of iterations, including the cases where complex values appeared. The maximum number of iterations (max-iter) was less than 150 in all cases, with much less than 30 in the majority of the datasets. The median number of iterations (med-iter) is less than 20 in the majority of datasets, showing that the Weiszfeld algorithm converges in a very short amount of time. The number of iterations for the indefinite kernels are in most cases only slightly larger than the values for positive definite kernels, showing that they converge equally well in practice. In all tested datasets, complex weights only appeared in a total of 30 cases using indefinite kernels, and – as expected – nowhere for positive definite kernels. This shows that even when using indefinite kernel functions, the weights remain real values in a large majority of the time.

In summary, the Weiszfeld algorithm converged in all cases we have tested, even though we cannot give a formal proof that it is always the case for indefinite kernels.

5.4 Experiments on the generalized median reconstruction

As discussed in Section 3, the weighted mean function can not guarantee that the computed approximated median is closer to the true generalized median than the previous objects. Nevertheless, one can assume that even if it is not always the case, using a large number of combinations leads to a better approximation than using a small number of weighted mean combinations. For this reason the linear recursive and triangular recursive reconstruction methods shown in Algorithm 2 will likely deliver a better approximation than linear and triangular methods shown in Algorithm 1.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

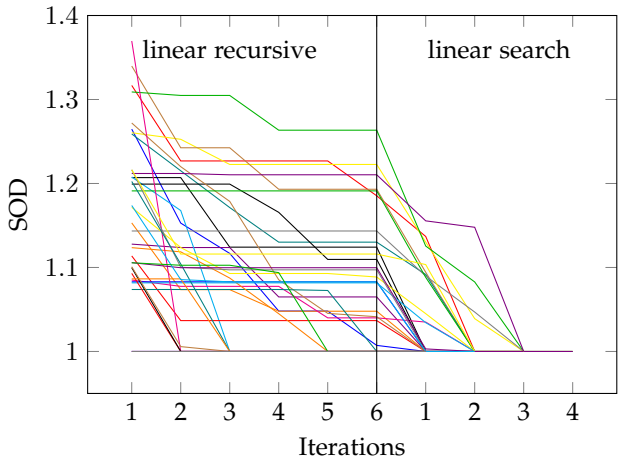


Figure 4. Minimization of the SOD on the Darwin dataset using linear-recursive (first 6 iterations) and linear search (second 4 iterations). Values are normalized by the final SOD.

Figure 4 shows the evolution of the sum of distances in the reconstruction using linear recursive reconstruction (Algorithm 2), followed by linear search (see Section 5.5) on the Darwin dataset (a similar evolution was observed on the other datasets). Each of these 36 lines (not all differential due to overlapping) represents one string set being minimized by these methods. All values in one reconstruction process have been normalized by their final sum of distances. As can be seen, even though the notion of projection is not entirely clear in the space of strings, the linear recursive reconstruction iteratively improves the computed median by combining strings with the weighted mean function, leading overall to a lower sum of distances over time. One iteration in this sense is the pairwise combination of all objects in the current set (Lines 2 to 11 in Algorithm 2). Linear search further improves this approximated median by performing a line search using the weighted mean between the computed approximated median and every object in the set. Note that linear search stops early when no further improvement could be made. Note that a monotone falling sum of distances of the best median object is ensured here and in all results of our evaluation by simply saving the previously best object and including it in the set of objects for the next iteration.

As can be seen, both methods successfully improve the sum of distances of the approximated median over time, even though it is not guaranteed to improve in every iteration due to the often not-unique nature of the weighted mean for objects. Nevertheless, there is a significant enough improvement in a wide range of datasets such that these methods can be used to reconstruct an approximate generalized median.

5.5 Median quality

Our kernel-based method was tested on the datasets presented in 5.1. In all cases, we compared our method using the appropriate distances with the traditional embedding method (either prototype or CCA embedding). CCA was shown [1] to consistently provide the best results and thus chosen as a representative for this comparison. As in [1], all explicit embedding methods use $0.8 \cdot n$ dimensions for the target vectors. The parameters of the domain-independent kernel functions are $\beta = 2$ for K_δ^{nd}, $\gamma = 1$ and $p = 1$ for K_δ^{pol}. For K_δ^{comb}, $|\bar{O}| = 3$ objects were chosen using a k-medians algorithm. These parameters were chosen to guarantee a distance-preserving embedding if possible, as discussed in Section 4.5.3.

For each dataset and method, the average of sum of distances results over all sets within the respective dataset is reported in Table 4, with the best result of each reconstruction method marked in bold. Note that the sixth row in each table, e.g. K^{ssk} for the Darwin dataset, is related to a kernel specific to the particular domain and thus varies in each case. Since the absolute SOD depends on the distance function and dataset, all values were normalized using the linear transform $\frac{x-LB}{LB}$, where LB is the lower bound of the generalized median computed with a linear programming method [67]. Using this, a result of 0 would be a result that is guaranteed to be an optimal generalized median, while a result of 1 would mean a result whose SOD is 100% larger than the lower bound. Therefore, a lower result means a better approximated generalized median. Note, however, that the true generalized median does not necessarily have a result of 0 since the lower bound may not be reached. The true generalized median can be any value greater than 0. For comparison purpose Table 5 shows the absolute values of the median SOD as well as the median of lower bounds (LB) over all sets of a dataset.

To further show the difference in median quality, we also measured if there is a statistically significant improvement using the presented kernel methods over the distance-preserving framework. Values marked with a plus (+) are significantly better than the previously best CCA embedding, while values marked with a star (*) have no significant difference to CCA according to the Wilcoxon signed rank test [68] with a p-value threshold of 0.05. The signed rank test measures if the distribution of differences in the results of two methods on the same data has a mean of zero or not. If it is not zero, then one method consistently performs better than the other.

In addition to the reconstruction methods discussed in Section 4.3, we also included the linear search method presented in [34]. This method improves given approximations of the generalized median by a local search using the weighted mean between the approximated result and the objects in the set. As a post-processing, it is independent of the basic reconstruction method and can be used with kernel methods without modification. In our case, the linear recursive result was used as starting point.

As can be seen in Table 4, kernel methods consistently outperform or match the performance of explicit embedding methods. In most datasets and reconstructions, the four distance-preserving kernels K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} show the best results, being statistically better than CCA in 68 of 120 combinations of kernel and reconstruction methods, and equally good in 48 of 120 cases, leaving only four cases where CCA is statistically better than these kernels (ranking-bio using linear reconstruction). K_δ^{rbf}, however, often fails to compute accurate median approximations.

Of the domain specific kernel functions, K^{part} shows the best results, especially with being equally good or better

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

than CCA in the UCI Cluster dataset. K^{kend}, however, shows results mostly worse than CCA with the exception of one case, and K^{ssk} consistently underperforms for all string datasets. This is consistent with the fact that K^{part} is the kernel function closest related to the distance function used in the domain, while K^{kend} only shares a tangential relationship to the respective distance and K^{ssk} which shares no similarity to the string edit distance.

This finding generally confirms the expectation awakened in Section 5.2 that a perfect distance preservation should ensure more accurate median computation, even with a large amount of approximations in the computation. As all kernels with a perfect distance preservation perform nearly equally well in all datasets, one can assume that the specific kernel method is not as important to the median approximation as the fact that it is distance-preserving. In addition, they perform better with reconstruction methods involving more objects, like linear recursive or triangular recursive, where the correct relationships after the embedding play an even greater role in the reconstruction.

In Table 6 the average rank of each method is listed. Here we ranked the absolute sum of distances results of each method for each dataset and domain for a reconstruction method using 1 for the best result, 2 for the second best, and so on, meaning that a method with continually better results than other methods will have a lower rank. Additionally, the last column shows the average rank of each kernel and explicit embedding method over all reconstruction methods. K_δ^{lin} and K_δ^{pol} consistently show the best rank for each reconstruction method with K_δ^{nd} being slightly better for linear and triangular reconstruction but worse for linear-recursive, triangular-recursive and linear search. With a rank of around 1.3 to 1.4 in total, K_δ^{nd} and K_δ^{pol} are consistently on the first place for each dataset, meaning that it is a good choice for median computation in general. This confirms the discussion made in section 5.2, that is, a better distance approximation in vector space leads to a better median reconstruction. Comparing the domain specific kernel functions, it can also be seen that a closer relationship of the kernel function to the used distance leads to a better median approximation. K^{part} and K^{kend}, while only being tangentially related to the respective distance functions, deliver a better result on average than K^{ssk} which has no relationship to the string edit distance at all.

5.6 Correlation with distance preservation

The relationship between our results and the distance-preserving properties of the kernels can be seen in Figure 5. In this figure, the Normalized Cross Correlation (NCC) between the distances in kernel space and the original distances is compared to the normalized sum of distances (SOD) result. The NCC measures how linearly dependent two sets of values are, and in this case, how closely Eq. (17) is approximated by one c. In a perfect embedding, the NCC would be 1, while it would be 0 if there is no linear relationship between the original distances and the ones in kernel or embedding space. As can be seen in Figure 5, a high NCC correlates with a good approximation of the generalized median, although with a few exceptions. For

Table 4
Median Quality on Six Datasets: Comparison with SOD. Values marked by a + (*) are statistically significantly better than (similar to) CCA.

(a) Darwin dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.3683***	**0.3090***	**0.2070+**	**0.1969***	0.1056*
K_δ^{nd}	**0.3683+**	**0.3090***	**0.2070+**	**0.1969***	0.1056*
K_δ^{pol}	**0.3683***	**0.3090***	**0.2070+**	**0.1969***	0.1056*
K_δ^{rbf}	0.8284	0.8284	0.8284	0.8284	0.1925
K_δ^{comb}	**0.3683+**	**0.3090***	**0.2070+**	0.2041*	0.1056*
K^{ssk}	0.4107	0.3682	0.4023	0.2932	0.1487
CCA	0.3691	0.3144	0.2425	0.1988	**0.0988**
Prototype	0.6411	0.5705	0.2357*	0.2912	0.1275

(b) CCD dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.3103***	**0.3024***	**0.2693+**	**0.2769***	**0.2330+**
K_δ^{nd}	**0.3103***	**0.3024***	**0.2693+**	**0.2769***	**0.2330+**
K_δ^{pol}	**0.3103***	**0.3024***	**0.2693+**	**0.2769***	**0.2330+**
K_δ^{rbf}	0.6376	0.6376	0.6376	0.6376	0.2948
K_δ^{comb}	**0.3103***	**0.3024***	**0.2693+**	**0.2769***	**0.2330+**
K^{ssk}	0.4513	0.4455	0.4829	0.4560	0.2970
CCA	0.3177	0.3177	0.2812	0.2835	0.2590
Prototype	0.4693	0.4241	0.3001	0.2944	0.2452*

(c) Gen. Cluster dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.4575***	**0.4567+**	**0.4412+**	0.4567+	**0.4075***
K_δ^{nd}	**0.4575***	**0.4567+**	**0.4412+**	0.4567*	0.4098*
K_δ^{pol}	**0.4575***	**0.4567+**	**0.4412+**	0.4567+	**0.4075***
K_δ^{rbf}	0.6968	0.6968	0.6968	0.6968	0.4560*
K_δ^{comb}	**0.4575***	**0.4567***	**0.4412+**	**0.4428+**	0.4098*
K^{part}	0.4583	0.4583	**0.4412***	0.4583	0.4098*
CCA	**0.4575**	**0.4567**	0.4575	0.4543	0.4183
Prototype	0.6055	0.5706	0.4966	0.4862	0.4176*

(d) UCI Cluster dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.3055+**	**0.3055+**	**0.3055***	**0.3055+**	**0.2844***
K_δ^{nd}	**0.3055+**	**0.3055+**	**0.3055***	**0.3055+**	**0.2844***
K_δ^{pol}	**0.3055+**	**0.3055+**	**0.3055***	**0.3055+**	**0.2844***
K_δ^{rbf}	1.4435	1.4435	1.4435	1.4435	0.2907*
K_δ^{comb}	**0.3055+**	**0.3055+**	**0.3055***	**0.3055+**	**0.2844***
K^{part}	**0.3055***	**0.3055***	**0.3055***	**0.3055***	**0.2844+**
CCA	0.3078	0.3078	0.3078	0.3078	0.2929
Prototype	0.4385	0.4301	0.3321	0.3427	0.2964*

(e) ranking-bio dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.1113**	**0.1113+**	**0.1113+**	**0.1113+**	**0.1058+**
K_δ^{nd}	**0.1113**	**0.1113+**	**0.1113+**	**0.1113+**	**0.1058+**
K_δ^{pol}	**0.1113**	**0.1113+**	**0.1113+**	**0.1113+**	**0.1058+**
K_δ^{rbf}	0.1458	0.1458	0.1458	0.1458	0.1324
K_δ^{comb}	**0.1113**	**0.1113+**	**0.1113+**	**0.1113+**	**0.1058+**
K^{kend}	0.1187	0.1187	0.1412	0.1187	0.1063
CCA	**0.1113**	**0.1113**	**0.1113**	**0.1113**	0.1087
Prototype	**0.1113**	**0.1113**	**0.1113**	**0.1113**	0.1087

(f) ranking-real dataset:

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	**0.2818+**	0.2818*	**0.2649+**	**0.2698+**	**0.2200+**
K_δ^{nd}	**0.2818+**	0.2818*	**0.2649+**	**0.2698+**	**0.2200+**
K_δ^{pol}	**0.2818+**	0.2818*	**0.2649+**	**0.2698+**	**0.2200+**
K_δ^{rbf}	0.5647	0.5647	0.5647	0.5647	0.2662
K_δ^{comb}	**0.2818+**	0.2818*	**0.2649+**	0.2743+	**0.2200+**
K^{kend}	0.3058	0.3058	0.2783	0.3058	0.2240*
CCA	0.2968	**0.2812**	0.2723	0.2831	0.2216
Prototype	0.3841	0.3826	0.3000	0.3203	0.2363

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

Table 5
Values Used for the Normalization of the Results

	median SOD	median LB
Darwin	755.00	619.50
CCD	1094.00	861.00
Gen-Cluster	891.00	606.50
UCI-Cluster	2823.00	2033.50
ranking-bio	2574.00	2304.00
ranking-real	2398.50	1850.00

Table 6
Median Quality on Six Datasets: Comparison with Ranking

(a) without domain-specific methods:

	linear	triangular	lin-rec	triang-rec	lin-search	total
K_δ^{lin}	1.12	**1.20**	1.38	**1.37**	**1.55**	**1.32**
K_δ^{nd}	**1.11**	**1.20**	**1.35**	1.41	1.58	1.33
K_δ^{pol}	1.12	**1.20**	1.38	**1.37**	**1.55**	**1.32**
K_δ^{rbf}	6.23	6.27	6.42	6.41	5.07	6.08
K_δ^{comb}	1.15	1.24	1.39	1.40	1.58	1.35
CCA	1.42	1.99	2.73	2.61	2.58	2.27
Prototype	5.04	5.19	4.18	4.54	3.77	4.54

(b) with domain-specific methods:

	linear	triangular	lin-rec	triang-rec	lin-search	total
K_δ^{lin}	1.18	1.27	1.43	**1.42**	**1.72**	**1.40**
K_δ^{nd}	**1.18**	**1.26**	**1.39**	1.45	1.74	1.40
K_δ^{pol}	1.18	1.27	1.43	**1.42**	**1.72**	**1.40**
K_δ^{rbf}	7.07	7.11	7.25	7.25	5.67	6.87
K_δ^{comb}	1.22	1.30	1.43	1.44	1.74	1.43
K^{ssk}	5.30	5.25	6.37	6.21	4.95	5.62
K^{part}	2.42	2.67	2.90	2.77	2.55	2.66
K^{kend}	3.37	4.09	4.60	4.39	3.25	3.94
CCA	1.51	2.05	2.81	2.66	2.78	2.36
Prototype	5.68	5.79	4.51	4.92	4.13	5.01

example in the String datasets, the prototype embedding is worse than K_δ^{rbf}, even with a much higher NCC value.

5.7 Computational time

All experiments were performed using a Intel Core i5-4590 (4 x 3.3GHz) with 16GB RAM using Python 3.5. Table 7 shows the average time of the median computation depending on the different reconstruction methods. As expected, the more objects are involved in the reconstruction, the more time is needed, independent of if kernel methods or traditional embedding methods are used. As most of the time in the computation is used in the reconstruction, one can see that – with the exception of linear search – the run time is near constant for each embedding variant. Note that CCA has a near constant base-time of approximate 4 seconds

Table 7
Mean Computational Time on the Darwin Dataset (in Seconds)

	linear	triangular	lin-rec	triang-rec	lin-search
K_δ^{lin}	0.09	0.15	2.79	2.83	49.54
K_δ^{nd}	0.08	0.15	2.57	2.70	49.38
K_δ^{pol}	0.10	0.17	2.84	2.74	49.74
K_δ^{rbf}	0.08	0.14	2.59	2.66	78.98
K_δ^{comb}	0.11	0.18	3.08	3.06	49.88
K^{ssk}	0.17	0.24	2.90	2.91	64.17
CCA	4.32	4.38	6.95	6.57	57.20
Prototype	0.07	0.13	2.74	2.61	58.09

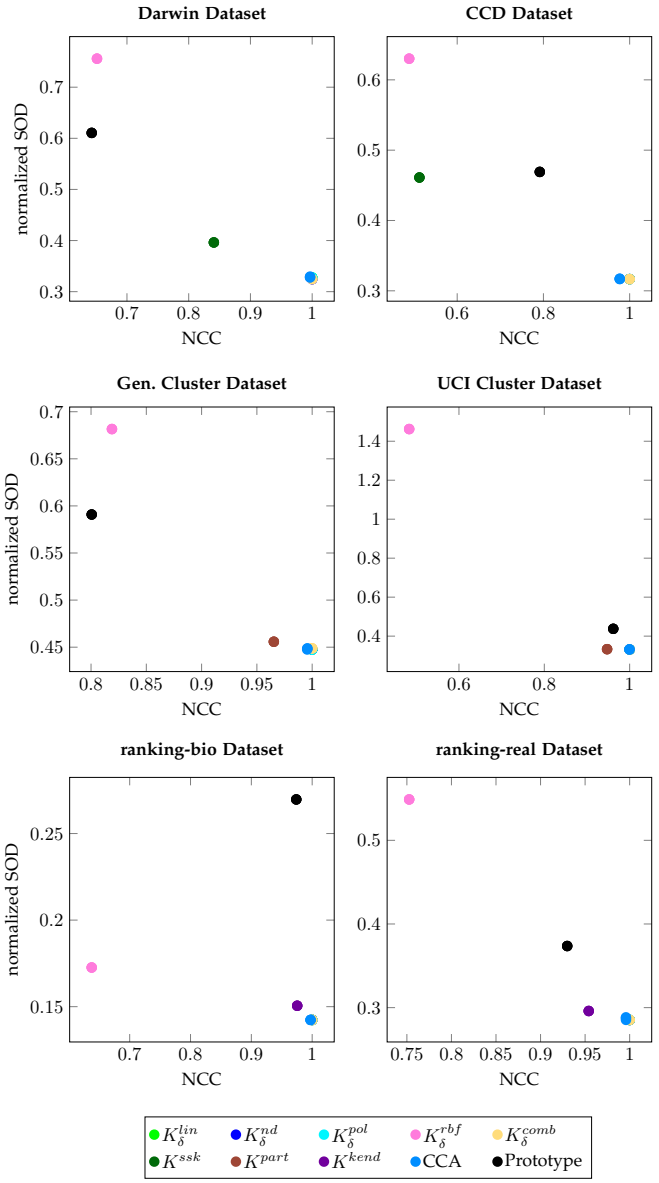


Figure 5. Relationship between embedding and median quality in the different datasets, measured in NCC and SOD respectively.

due to the computational time of the much more complex initial embedding in each reconstruction method. Although computing the Weiszfeld algorithm using kernel functions is more complex than using embedded vectors, the fast convergence only causes very little additional time compared to Prototype embedding using the original Weiszfeld algorithm. For linear search however, the methods display a larger variance in computational time. This can be explained by the nature of this reconstruction method. In contrast to the previous ones, linear search uses the result of linear-recursive as starting point and stops early once convergence is reached, meaning that due to the better base results, K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} require less iterations of linear search than K_δ^{rbf}, K^{ssk}, CCA and Prototype. As such, using K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} not only improves the embedding quality, but also the required time for the linear-search reconstruction. In particular, K_δ^{rbf} initially shows the

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

worst sum of distances results by a large margin due to the high distortion of distances, which leads to a much higher required run-time of linear search.

In summary, we expect K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} to have a lower computational time than CCA in all cases. All of these kernel methods can be easily computed from the distance function at hand, leading to fast kernel computations that preserve distances in the embedded space. CCA requires all pairwise distances for its embedding as well, but additional time for the computation of embedding vectors. Especially for large datasets and non-trivial embeddings, this can require a large number of iterations that are not present in distance based kernel methods. In contrast to the difference in embedding, the kernel Weiszfeld method is only barely more computationally complex as the vector-based Weiszfeld algorithm. In combination with the sum of distances results, it is therefore recommended to use distance preserving kernel methods for median approximation instead of explicit embedding methods like CCA or Prototype.

6 Toolbox

We have implemented a toolbox in Python that contains a large number of embedding-based methods for consensus learning, including the most recent development of distance-preserving embedding method [1] and our current work of kernel-based method. The user can apply these techniques on any dataset, provided a distance function and weighted mean function between two objects of the set are made available. It includes, among others, a variety of embedding, kernel, and reconstruction methods. It is implemented in such a modular manner that the user can easily insert additional methods and functions. In addition to embedding-based methods, the toolbox also contains other methods for generalized median computation such as the evolutionary weighted mean based framework [31]. In the toolbox part of the functions is also available in MATLAB. By providing the toolbox for public use (available at: http://pria.uni-muenster.de), we encourage other researchers to explore generalized median computation and applications, especially for those problem instances with high inherent computational complexity.

7 Conclusion

In this paper we have shown how the generalized median can be computed using kernel methods, without using an explicit embedding like in previous methods. Although positive definite kernels would be preferred for the implicit embedding into an Euclidean vector space, we showed how indefinite kernels can also be used as an implicit embedding into a pseudo-Euclidean vector space. On six datasets, regularly better results could be demonstrated than methods using explicit embedding. Kernel-based generalized median computation not only shows overall superior results, but also overcomes the inherent drawbacks of current generalized median computation using explicit transformation. From the studied kernels, distance-preserving kernels K_δ^{lin}, K_δ^{nd}, K_δ^{pol} and K_δ^{comb} have, as expected, the best results, confirming the correlation between distance preservation and generalized median result. One should therefore use one of these kernel functions as baseline for median computation in practice.

Our goal in this work is to introduce a way to compute the generalized median for arbitrary spaces (with the weak requirement of weighted mean) and to demonstrate its good performance. As such, it is difficult to give theoretical guarantees for the quality of the generalized median in general. In fact, research papers in the literature with theoretical guarantees are typically done for a specific space explicitly using concrete knowledge of the space. Instead, we demonstrated superior performance compared to the previous distance-preserving and prototype embedding methods that represent the state of the art methods. For the comparison we used common datasets as in the publications of the previous methods, which can thus be considered as benchmark datasets. Overall, this performance comparison with state of the art methods on benchmark datasets helps to alleviate the lacking theoretical guarantees.

Although in our experiments the Weiszfeld algorithm converges in all cases, we were not be able to formally prove its convergence in the general case of indefinite kernels. This will be further studied in future. Aside from the computation of the median of a set, the kernel methods in this work can also applied to a number of other problems for more accurate computation. One example is kernel k-means, where a set of vectors is clustered in a kernel space to allow for non-convex clusters in the k-means algorithm [69]. One could use the median computation from this paper to allow not only clustering of non-vector data, but also using the median instead of the mean.

Acknowledgements

We sincerely thank the reviewers and the handling associate editor for their valuable comments that have substantially reshaped our paper. We thank Florian Eilers for the valuable discussion about complex vector spaces. This work was partly supported by the Deutsche Forschungsgemeinschaft (DFG) – CRC 1450 – 431460824 and the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 778602 Ultracept.

References

[1] A. Nienkötter and X. Jiang, "Distance-preserving vector space embedding for consensus learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 1244–1257, 2021.

[2] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[3] S. C. Boulakia, A. Denise, and S. Hamel, "Using medians to generate consensus rankings for biological data," in *Proc. of 23rd Int. Conf. on Scientific and Statistical Database Management (SSDBM)*, 2011, pp. 73–90.

[4] K. Rothaus, X. Jiang, and M. Lambers, "Comparison of methods for hyperspherical data averaging and parameter estimation," in *Proc. of 18th Int. Conf. on Pattern Recognition*, 2006, pp. 395–399.

[5] M. Storath and A. Weinmann, "Fast median filtering for phase or orientation data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 639–652, 2018.

[6] H. Ding, R. Berezney, and J. Xu, "k-prototype learning for 3D rigid structures," in *Proc. of NIPS*, 2013, pp. 2589–2597.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

[7] R. I. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, 2013.

[8] S. H. Lee and J. Civera, "Robust single rotation averaging," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2004.00732

[9] S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 3, pp. 337–372, 2011.

[10] C. Carpineto and G. Romano, "Consensus clustering based on a new probabilistic rand index with application to subtopic retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2315–2326, 2012.

[11] T. Boongoen and N. Iam-On, "Cluster ensembles: A survey of approaches with recent extensions and applications," *Computer Science Review*, vol. 28, pp. 1–25, 2018.

[12] H. Ding and J. Xu, "Finding median point-set using earth mover's distance," in *Proc. of 28th AAAI Conf. on Artificial Intelligence (AAAI)*, 2014, pp. 1781–1787.

[13] B. Berkels, G. Linkmann, and M. Rumpf, "An SL(2) invariant shape median," *Journal of Mathematical Imaging and Vision*, vol. 37, no. 2, pp. 85–97, 2010.

[14] Z. Wu, X. Jiang, N. Zheng, Y. Liu, and D. Cheng, "Exact solution to median surface problem using 3d graph search and application to parameter space exploration," *Pattern Recognition*, vol. 48, no. 2, pp. 380–390, 2015.

[15] L. Franek, D. D. Abdala, S. Vega-Pons, and X. Jiang, "Image segmentation fusion using general ensemble clustering methods," in *Proc. of 10th Asian Conf. on Computer Vision (ACCV), Part IV*, 2010, pp. 373–384.

[16] X. Jiang, J. Wentker, and M. Ferrer, "Generalized median string computation by means of string embedding in vector spaces," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 842–852, 2012.

[17] A. Nienkötter and X. Jiang, "Consensus learning for sequence data," in *Data Mining in Time Series and Streaming Databases*, M. Last, H. Bunke, and A. Kandel, Eds. World Scientific, 2018, pp. 69–91.

[18] X. Jiang, A. Münger, and H. Bunke, "On median graphs: properties, algorithms, and applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1144–1151, 2001.

[19] L. Mukherjee, V. Singh, J. Peng, J. Xu, M. J. Zeitz, and R. Berezney, "Generalized median graphs and applications," *Journal of Combinatorial Optimization*, vol. 17, no. 1, pp. 21–44, 2009.

[20] Y. Xie, J. Ho, and B. C. Vemuri, "Multiple atlas construction from a heterogeneous brain MR image collection," *IEEE Trans. on Medical Imaging*, vol. 32, no. 3, pp. 628–635, 2013.

[21] S. Hauberg, A. Feragen, and M. J. Black, "Grassmann averages for scalable robust PCA," in *Proc. of CVPR*, 2014, pp. 3810–3817.

[22] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, and H. Bunke, "Generalized median graph computation by means of graph embedding in vector spaces," *Pattern Recognition*, vol. 43, no. 4, pp. 1642–1655, 2010.

[23] Y. Lim and M. Pálfia, "Matrix power means and the Karcher mean," *Journal of Functional Analysis*, vol. 262, no. 4, pp. 1498–1514, 2012.

[24] W. Tang, "Mallows ranking models: maximum likelihood estimate and regeneration," in *Int. Conf. on Machine Learning*, 2019, pp. 6125–6134.

[25] C. de la Higuera and F. Casacuberta, "Topology of strings: Median string is NP-complete," *Theoretical Computer Science*, vol. 230, no. 1, pp. 39–48, 2000.

[26] M. Krivánek and J. Morávek, "*NP*-hard problems in hierarchical-tree clustering," *Acta Informatica*, vol. 23, no. 3, pp. 311–323, 1986.

[27] M. Bader, "The transposition median problem is NP-complete," *Theoretical Computer Science*, vol. 412, no. 12-14, pp. 1099–1110, 2011.

[28] S. L. Hakimi, "Location theory," *Rosen, Michaels, Gross, Grossman, Shier (eds.) Handbook Disc. & Comb. Math. CRC Press, Boca Raton*, 2000.

[29] C. Carpineto and G. Romano, "Optimal meta search results clustering," in *Proc. 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2010, pp. 170–177.

[30] N. Boria, S. Bougleux, B. Gaüzère, and L. Brun, "Generalized median graph via iterative alternate minimizations," in *Proc. of Int. Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 2019, pp. 99–109.

[31] L. Franek and X. Jiang, "Evolutionary weighted mean based framework for generalized median computation with application to strings," in *Proc. of IAPR Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, 2012, pp. 70–78.

[32] M. Ferrer, D. Karatzas, E. Valveny, I. Bardají, and H. Bunke, "A generic framework for median graph computation based on a recursive embedding approach," *Computer Vision and Image Understanding*, vol. 115, no. 7, pp. 919–928, 2011.

[33] L. Franek and X. Jiang, "Ensemble clustering by means of clustering embedding in vector spaces," *Pattern Recognition*, vol. 47, no. 2, pp. 833–842, 2014.

[34] A. Nienkötter and X. Jiang, "Improved prototype embedding based generalized median computation by means of refined reconstruction methods," in *Proc. of Joint IAPR Int. Workshop on Structural, Syntactic, and Statistical Pattern Recognition (S+SSPR)*. Springer, 2016, pp. 107–117.

[35] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Reconstructing the house from the ad: Structured prediction on real estate classifieds," in *Proc. of 15th Conf. of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 274–279.

[36] A. Korba, A. Garcia, and F. d'Alché Buc, "A structured prediction approach for label ranking," in *Proc. of NeurIPS*, 2018, pp. 9008–9018.

[37] M. M. Arzani, M. Fathy, A. A. Azirani, and E. Adeli, "Switching structured prediction for simple and complex human activity recognition," *IEEE Trans. on Cybernetics*, vol. 51, no. 12, pp. 5859–5870, Dec. 2021.

[38] X. L. Liao, C. Zhang, M. Dong, and X. Chen, "Person reidentification by deep structured prediction — A fully parameterized approach," *IEEE MultiMedia*, vol. 26, no. 3, pp. 42–55, Jul. 2019.

[39] E. B. Wilson, "First and second laws of error," *Journal of the American Statistical Association*, vol. 18, no. 143, pp. 841–851, 1923.

[40] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance*. Springer Science & Business Media, 2012.

[41] D. Hsu, "Long-tailed distributions for position errors in navigation," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 62–72, 1979.

[42] R. Arora and M. Meilă, "Consensus ranking with signed permutations," in *Proc. of 16th Int. Conf. on Artificial Intelligence and Statistics*, 2013, pp. 117–125.

[43] C. Martinez-Hinarejos, A. Juan, and F. Casacuberta, "Improving classification using median string and nn rules," in *Proc. of National Symposium on Pattern Recognition and Image Analysis*, 2001.

[44] E. Weiszfeld and F. Plastria, "On the point for which the sum of the distances to n given points is minimum," *Annals of Operations Research*, vol. 167, no. 1, pp. 7–41, 2009.

[45] P. Honeine and C. Richard, "Preimage problem in kernel-based machine learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77–88, 2011.

[46] A. Beck and S. Sabach, "Weiszfeld's method: Old and new results," *Journal of Optimization Theory and Applications*, vol. 164, no. 1, pp. 1–40, 2015.

[47] H. W. Kuhn, "A note on Fermat's problem," *Mathematical Programming*, vol. 4, no. 1, pp. 98–107, 1973.

[48] H. Bunke and S. Günter, "Weighted mean of a pair of graphs," *Computing*, vol. 67, no. 3, pp. 209–224, 2001.

[49] H. Bunke, X. Jiang, K. Abegglen, and A. Kandel, "On the weighted mean of a pair of strings," *Pattern Analysis and Applications*, vol. 5, no. 1, pp. 23–30, 2002.

[50] L. Franek, X. Jiang, and C. He, "Weighted mean of a pair of clusterings," *Pattern Analysis and Applications*, vol. 17, no. 1, pp. 153–166, 2014.

[51] S. Ma and M. Belkin, "Kernel machines that adapt GPUs for effective large batch training," in *Proc. of 2nd SysML Conference*, 2019.

[52] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[53] T. Gärtner, "A survey of kernels for structured data," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 49–58, 2003.

[54] ——, *Kernels for Structured Data*. World Scientific, 2008.

[55] L. Xu, X. Jiang, L. Bai, J. Xiao, and B. Luo, "A hybrid reproducing graph kernel based on information entropy," *Pattern Recognition*, vol. 73, pp. 89–98, 2018.

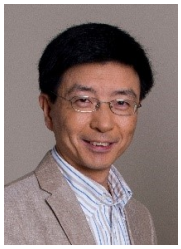
[56] S. Ghosh, N. Das, T. Gonçalves, P. Quaresma, and M. Kundu, "The journey of graph kernels through two decades," *Computer Science Review*, vol. 27, pp. 88–111, 2018.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3202565

[57] B. Haasdonk, "Feature space interpretation of svms with indefinite kernels," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 482–492, 2005.
[58] E. Pekalska, P. Paclik, and R. P. Duin, "A generalized kernel approach to dissimilarity-based classification," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 175–211, 2001.
[59] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.
[60] Y. Jiao and J.-P. Vert, "The Kendall and Mallows kernels for permutations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 7, pp. 1755–1769, 2017.
[61] B. Haasdonk and C. Bahlmann, "Learning with distance substitution kernels," in *Pattern Recognition*, C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, Eds. Springer, 2004, pp. 220–227.
[62] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recognition*, vol. 39, no. 10, pp. 1852 – 1863, 2006.
[63] C. Lundsteen, J. Phillip, and E. Granum, "Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes," *Clinical Genetics*, vol. 18, pp. 355–370, 1980.
[64] K. Bache and M. Lichman, "UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences," 2013.
[65] B. Brancotte, B. Yang, G. Blin, S. Cohen-Boulakia, A. Denise, and S. Hamel, "Rank aggregation with ties: Experiments and analysis," *Proc. of the VLDB Endowment*, vol. 8, no. 11, pp. 1202–1213, 2015.
[66] H. Üster and R. Love, "The convergence of the Weiszfeld algorithm," *Computers & Mathematics with Applications*, vol. 40, no. 4-5, pp. 443–451, 2000.
[67] X. Jiang and H. Bunke, "Optimal lower bound for generalized median problems in metric space," in *Proc. of Joint IAPR Int. Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer, 2002, pp. 143–151.
[68] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[69] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proc. of 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2004, pp. 551–556.

Andreas Nienkötter received the master's degree in computer science from University of Münster, Münster, Germany, in 2015, followed by a Ph.D. in 2021 at the same university. He is currently in a Postdoc position in Prof. X. Jiang's Research Group for Pattern Recognition and Image Analysis. His research interests include consensus learning using the generalized median, vector space embedding methods, and dimensionality reduction methods.



Xiaoyi Jiang received the bachelor's degree from Peking University, Beijing, China, and the Ph.D. and Venia Docendi (Habilitation) degrees from the University of Bern, Bern, Switzerland, all in Computer Science. He was an Associate Professor with the Technical University of Berlin, Berlin, Germany. Since 2002, he has been a Full Professor of Computer Science with the University of Münster, Münster, Germany, where he is currently the Dean of the Faculty of Mathematics and Computer Science. His current research interests include biomedical imaging, 3D image analysis, and structural pattern recognition. Dr. Jiang is an Editor-in-Chief of International Journal of Pattern Recognition and Artificial Intelligence. He also serves on the Advisory Board and Editorial Board of several journals, including International Journal of Neural Systems and Journal of Big Data. Previously, he has been Associate Editor for IEEE Trans. on Systems, Man, and Cybernetics - Part B / IEEE Trans. on Cybernetics, IEEE Trans. on Medical Imaging, and Pattern Recognition. He is Chair of IEEE EMBS Technical Committee on Biomedical Imaging and Image Processing (BIIP). He is a Senior Member of IEEE and a Fellow of IAPR.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.