

Improved Prototype Embedding Based Generalized Median Computation by Means of Refined Reconstruction Methods

Andreas Nienkötter and Xiaoyi Jiang^(✉)

Department of Mathematics and Computer Science,
University of Münster, Münster, Germany
xjiang@uni-muenster.de

Abstract. Learning a prototype from a set of given objects is a core problem in machine learning and pattern recognition. A popular approach to consensus learning is to formulate it as an optimization problem in terms of generalized median computation. Recently, a prototype-embedding approach has been proposed to transform the objects into a vector space, compute the geometric median, and then inversely transform back into the original space. This approach has been successfully applied in several domains, where the generalized median problem has inherent high computational complexity (typically \mathcal{NP} -hard) and thus approximate solutions are required. In this work we introduce three new methods for the inverse transformation. We show that these methods significantly improve the generalized median computation compared to previous methods.

1 Introduction

Learning a prototype from a set of given objects is a core problem in machine learning and pattern recognition, and has numerous applications [4, 6, 10]. One often needs a representation of several similar objects by a single consensus object. One example is multiple classifier combination for text recognition, where a change in algorithm parameters or the use of different algorithms can lead to distinct results, each with small errors. Consensus methods produce a text which best represents the different results and thus removes errors and outliers.

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

$$\bar{o} = \arg \min_{o \in \mathcal{O}} SOD(o) \quad (1)$$

where SOD is the sum of distances

$$SOD(o) = \sum_{p \in O} \delta(o, p).$$

In other words, the generalized median is an object which has the smallest sum of distances to all input objects. Note that the median object is not necessarily part of set O .

The concept of generalized median has been studied for numerous problem domains related to a broad range of applications, for instance strings [10] and graphs [9]. As the generalized median is not necessary part of the set, any algorithm must be of constructive nature and the construction process crucially depends on the structure of the objects under consideration. In addition, the generalized median computation is provably of high computational complexity in many cases. For instance, the computation of generalized median string turns out to be \mathcal{NP} -hard [8] for the string edit distance. The same applies to median ranking under the generalized Kendall- τ distance [3] and ensemble clustering for reasonable clustering distance functions, e.g. the Mirkin-metric [11].

Given the high computational complexity, approximate solutions are required to calculate the generalized median in reasonable time. Recently, a prototype embedding approach has been proposed [5], which is applicable to any problem domain and has been successfully applied in several domains [4–6, 10]. In this work we investigate the reconstruction issue of this approach towards further improvement of consensus learning quality.

In the prototype embedding approach, the objects are embedded into an Euclidean metric space using the embedding function

$$\varphi(o) = (\delta(o, p_1), \delta(o, p_2), \dots, \delta(o, p_d)) \quad (2)$$

where $\delta()$ is a distance between two objects. This embedding function assigns each object o_i to a vector $x_i = \varphi(o_i)$, which consists of its distance to d selected prototype objects $p_1, \dots, p_d \in O$. Different methods were suggested for the prototype selection, for example k-means clustering, border object selection and others [2]. Using the vectors x_i , the geometric median in vector space is calculated by the Weiszfeld algorithm [16]. Then, the geometric median is transformed back into the original problem space, resulting in the searched generalized median.

In this work we propose new reconstruction methods to use with prototype embedding. A comparison with several established methods is conducted to demonstrate that the proposed new reconstruction methods can significantly improve the quality of generalized median computation (in terms of the optimization function SOD).

The remainder of the paper is organized as follows. In the next section the prototype embedding approach is briefly summarized. In Sect. 3 we then present various reconstruction methods proposed in this work. Experimental results are reported in Sect. 4. Finally, we conclude our findings in Sect. 5.

2 Prototype Embedding Based Generalized Median Computation

The vector space embedding approach consists of three steps (Fig. 1):

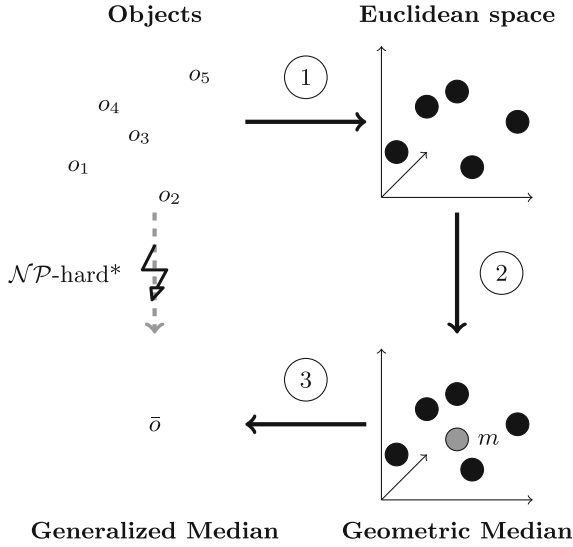


Fig. 1. Overview of prototype embedding based on [5]. *As mentioned in Sect. 1, the generalized median computation is not in all cases \mathcal{NP} -hard but in many domains with well-known metrics.

1. Embed the objects into a d -dimensional Euclidean space.
2. Compute the geometric median of the embedded points.
3. Estimate (reconstruct) the generalized median of the objects using an inverse transformation from the geometric median back into the original problem space.

2.1 Embedding Function

Prototype embedding uses (2) to embed objects into vector space. Here two design issues must be considered: the number d of selected prototypes and a selection algorithm. In our work the former issue is considered as a parameter, which will be systematically studied. For prototype selection, the so-called k-medians prototype selector turns out to be a good choice [5, 10]. It finds d clusters in the given set of data using the k-means clustering algorithm and declares the object with the smallest sum of distances in each cluster to be a prototype. In this work we thus will use this prototype selection method for the experiments.

2.2 Computation of Geometric Median by Weiszfeld Algorithm

Once the embedded points are determined, the median vector has to be computed. No algorithm is known for exactly computing the Euclidean median in polynomial time, nor has the problem been shown to be \mathcal{NP} -hard [7]. The most

common approximate algorithm is due to Weiszfeld [16]. Starting with a random vector $m_0 \in \mathbb{R}^d$, this algorithm converges to the desired median using the iterative function

$$m_{i+1} = \left(\sum_{j=1}^n \frac{x_j}{\|x_j - m_i\|} \right) / \left(\sum_{j=1}^n \frac{1}{\|x_j - m_i\|} \right)$$

2.3 Reconstruction Methods

The last step is to transform the Euclidean median from the vector space back into the original space. In the following we summarize three established strategies for this purpose (full details can be found in [4,5]). The foundation of all these heuristic approaches is the concept of weighted mean. The weighted mean \bar{o} between objects $o, p \in \mathcal{O}$ with ratio $0 \leq \alpha \leq 1$ is defined as

$$\delta(o, \bar{o}) = \alpha \cdot \delta(o, p), \quad \delta(\bar{o}, p) = (1 - \alpha) \cdot \delta(o, p)$$

In other words, the weighted mean is a linear interpolation between both objects. In many cases, this weighted mean function can be derived from the distance function between objects [4,6].

Linear Reconstruction. Linear reconstruction uses the two objects o_1 and o_2 associated with the two closest points to the geometric median. The geometric median m is projected onto the line between these points using a simple line projection, resulting in a ratio α . The generalized median in the original space \mathcal{O} is then calculated as the weighted mean between o_1 and o_2 using α .

Triangular Reconstruction. The triangulation method first finds the three closest points to the Euclidean median (denoted by x_1, x_2, x_3). Then, the median vector of these points is computed and denoted with x' . Now the method seeks to estimate an object in the original space \mathcal{O} which corresponds to point x' and is thus assumed to be an approximation of the generalized median. Two steps have to be conducted. First, two out of the three points are arbitrarily chosen. Without loss of generality we assume that x_1 and x_2 are selected and we then project the remaining point x_3 onto the line joining x_1 and x_2 by using x' . As a result a point m is received, lying between x_1 and x_2 . In this situation the linear interpolation method can be used to recover an object, which corresponds to m . A further application of the interpolation method then yields an object which corresponds to x' .

Recursive Reconstruction. The recursive reconstruction method [4] is more complex, in which the median is recursively projected onto hyperplanes of decreasing dimensionality. The median in n dimensions is calculated as a weighted mean between the median in $n - 1$ dimensions and the n -th object.

This results in a triangular or linear reconstruction if the recursion arrives at dimension three or two. Since this method calculates a generalized median in each step, one can use the intermediate result with the best sum of distances as the final result. This is the best-recursive interpolation.

3 Refined Reconstruction Methods

In this section we present several refined reconstruction methods that can be used in the framework described above towards more accurate generalized median computation.

3.1 Linear Recursive

In linear reconstruction, only the two nearest neighbors of the median vector are considered for the reconstruction. While this is fast and easily done, using only two objects restricts the search for a median object to the weighted mean between these objects. Therefore, we propose to consider the weighted mean between other objects as well. In linear recursive reconstruction, not only the nearest pair of neighbors is considered, but also the next nearest pair and all following ones. The objects are paired as $(o_1, o_2), (o_3, o_4), \dots, (o_{n-1}, o_n)$ with $\delta_e(m, \varphi(o_i)) \leq \delta_e(m, \varphi(o_{i+1})), \forall 1 \leq i < n$. In this way, the two nearest neighbors are selected as a pair, then the two next neighbors etc. If n is odd, then the last object is not processed in the current iteration, but directly taken over for the next iteration. Each pair is linearly reconstructed (see Sect. 2.3), resulting in $n/2$ median objects. This process is repeated using only the reconstructed objects, resulting in $n/4$ objects. After $\log(n)$ steps, only one object remains. The object with the best SOD from all intermediate results is returned as the generalized median. Naturally, this reconstruction method is at least as good as the linear reconstruction, since it is included as the first pair.

An example using four objects can be found in Fig. 2. The median vector m is first projected on lines connecting each pair of objects. Then, the median is recursively projected on lines connecting pairs of these projections.

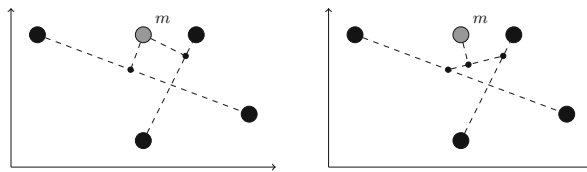


Fig. 2. Linear recursive reconstruction. Result after the first (left) and second iteration (right).

3.2 Triangular Recursive

Triangular recursive reconstruction is very similar to linear recursive reconstruction. The difference is that objects are grouped in triplets instead of pairs, using the triangular reconstruction (Sect. 2.3) to generate $n/3$ objects as a result. This is again repeated until only one object remains. For cases in which only two objects are present, linear reconstruction is used instead.

3.3 Linear Search

The above mentioned reconstruction methods calculate an approximation of the generalized median from the geometric median in vector space. Linear search improves this result by searching for a better approximation. It is reasonable to expect that a true generalized median is very similar to the calculated approximate one. Therefore, the region around the prior result should be searched for an object with lower SOD, see Fig. 3. Using weighted means (black dots) between the previous calculated median object \bar{o} and objects in the set, an object with a lower SOD is searched (red dot). This process is repeated with the new object, until the change in SOD is sufficiently small. Since the generalized median serves as a lower bound of the sum of distance and this method reduces the SOD in each step if possible, it eventually converges to an optimal solution. Using this method only lines between given objects are searched instead of the full neighborhood of \bar{o} . Therefore, it is not guaranteed to arrive at the true generalized median and could result in a local minimum.

The linear search is a refinement performed the original domain \mathcal{O} . It can be combined with any reconstruction method, which calculates an approximation of the generalized median from the geometric median in vector space.

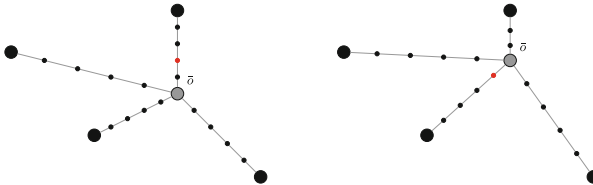


Fig. 3. Linear search reconstruction. This refinement search is performed the original domain \mathcal{O} . (Color figure online)

4 Experimental Evaluation

In this section we present experimental results. These results were generated using four datasets of two different types. First, we describe the datasets we used in our evaluation. Then, we show the results using our methods followed by a discussion of these results.

4.1 Datasets

Our methods were evaluated using four datasets in total, each consisting of several sets of objects. Table 1 shows some information about the used datasets.

Table 1. Evaluated datasets

Dataset	Type	Number of sets	Number of 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

The first two datasets consist of strings. The Darwin dataset is artificially generated with lines from Charles Darwin's famous work "On the Origin of Species". Each set of 40 strings was created from one line of length between 70 and 140 symbols, which was randomly distorted using probabilities based on real world data [10]. The Copenhagen Chromosome Dataset (CCD) includes 22 sets of genetic strings created by Lundsteen *et al.* [12]. Each string encodes selected parts of a chromosome. Each set consists of 100 strings with lengths from about 20 to 100 symbols. In both cases, the Levenshtein distance [15] is used.

The second two datasets contain clusters, encoded as label vectors. The first sets are clusters generated with data from the UCI Data Repository [1] using k-means clustering with different parameters. The second are artificially generated clusters made with the method of [6]. This method uses a cluster to generate several new ones by the substitution of labels. For the cluster datasets, we use the partition distance [6]. This metric expresses how many labels must be changed to result in identical partitions of the data. The labels themselves are not relevant.

4.2 Quality of Computed Generalized Median

Tables 2 and 3 show the relative sum of distances of the generalized median acquired using the different datasets. Since the maximum number of prototypes is dependent on the number of objects in each set, we used $d = 0.1n, 0.2n, \dots, 1n$. Table 2 displays the results of the different methods averaged over all tested d . Table 3 only includes the results of the best number of prototypes for each reconstruction method. The sets were embedded two times for each combination of number of prototypes and reconstruction method. The resulting SODs were averaged to compensate for small random influences in the results. The linear search method is combined with linear, best-recursive, and linear recursive methods as base result, respectively.

Table 2. Relative SOD results - average result

		Cluster	Gen. Clusters	CCD	Darwin
previous methods	linear	1.000	1.000	1.000	1.000
	triangular	1.005	0.835	0.972	0.946
	recursive	1.006	0.697	0.856	0.812
	best-recursive	0.904	0.674	0.760	0.712
proposed methods	linear-recursive	0.840	0.239	0.612	0.520
	triangular-recursive	0.865	0.262	0.601	0.518
	linear-search, linear	0.852	0.291	0.682	0.484
	linear-search, best-recursive	0.816	0.279	0.572	0.402
	linear-search, linear-recursive	0.785	0.033	0.533	0.359
	Linear Reconstruction SOD	4857.29	842.82	2799.27	951.65
	Lower Bound SOD	3429.25	586.00	1875.95	641.81

Table 3. Relative SOD results - best result

		Cluster	Gen. Clusters	CCD	Darwin
previous methods	linear	1.000	1.000	1.000	1.000
	triangular	1.002	0.648	0.969	0.902
	recursive	0.964	0.625	0.824	0.675
	best-recursive	0.906	0.625	0.725	0.592
proposed methods	linear-recursive	0.856	0.078	0.646	0.512
	triangular-recursive	0.875	0.126	0.605	0.506
	linear-search, linear	0.845	0.133	0.666	0.443
	linear-search, best-recursive	0.807	0.063	0.574	0.326
	linear-search, linear-recursive	0.805	0.000	0.563	0.327
	Linear Reconstruction SOD	4783.44	773.16	2710.48	899.72
	Lower Bound SOD	3429.25	586.00	1875.95	641.81

As the generalized median minimizes the SOD, a lower result means a more closely approximated result. Since the SOD is highly dependent on the distance function and dataset, a linear transformation $\frac{x-LB}{LR-LB}$ was used to normalize the results x and make them more easily comparable. The result (LR) using linear reconstruction is transformed to 1, the lower bound (LB) of the SOD of the generalized median is transformed to 0. As such, a value of 0.5 means that the calculated generalized median has a SOD which is at half between the result of prototype embedding with linear reconstruction and the lower bound. The lower bound was computed using the method from [9]. Since it may not be a very tight lower bound, it is not guaranteed that the generalized median would really produce a result of zero with this normalization.

The new reconstruction methods described in Sect. 3 show consistent and significant improvements of the SOD compared to previous methods both on average and their best dimension parameter, but the extent of the improvement is dependent on the chosen dataset. Linear-recursive and triangular-recursive

reconstruction have similar results, with one resulting in a lower SOD in Cluster datasets, and the other showing better results in string datasets. This indicates influences of the distance and weighted mean functions on the result. As expected, linear search methods improve their respective base results by a large margin. Moreover, the better the base object is, the better the resulting approximated generalized median is. One should therefore use linear recursive as a basis for this method.

4.3 Computational Time

For the experiments we used a computer with Intel Core i5-4590 (4×3.3 GHz) and 16 GB RAM, running Matlab 2015a in Ubuntu 15.10. Figure 4 shows the run time for the UCI Cluster dataset. Since the same methods were used to calculate the embedding and geometric median, only the time of each reconstruction method is included. The results of the other datasets are not shown due to their similarities to these results.

As expected, linear and triangular reconstruction are the fastest methods because only two or three weighted means are used in their computation. The number of dimensions is nearly irrelevant, because only very few distance calculations are needed. Recursive and best-recursive reconstruction use a number of objects equal to the embedding methods, which results in their linear runtime. Interestingly, the need to calculate the SOD for each intermediate result has no negative impact on the speed of the best-recursive approach to the recursive one. Linear recursive and triangular recursive use all objects of the set regardless of the embedding methods, as can be seen by their constant runtime. Linear search methods show the longest runtime, but also one that is nearly independent of the embedding dimension. Although the proposed reconstruction methods are on average slower than the traditional techniques, their significant improvement of the calculated median justifies their use.

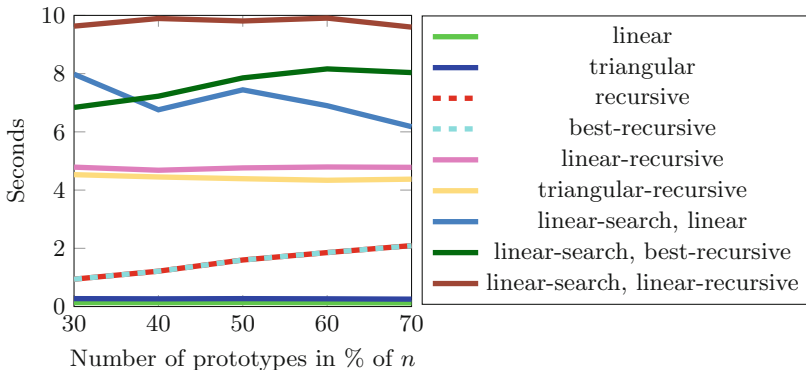


Fig. 4. Run-time of the UCI Cluster dataset.

5 Conclusion

Prototype embedding is a standard method for the embedding of arbitrary objects into Euclidean space with many applications and often used as an integral part of generalized median computation. In this work, we introduce new reconstruction methods for improving the generalized median computation. A study was presented using these methods on several datasets. The proposed linear recursive, triangular recursive, and linear search methods have been shown to consistently improve the results. Especially linear search using linear recursive reconstruction results in a significant improvement for all tested datasets. Although these results are derived from the datasets used in this work, our study provides hint enough to conclude that when dealing with generalized median computation we in general should take these reconstruction methods into account.

This work focused on the influence of reconstruction methods on the generalized median computation. Another point of interest is the use of alternative embedding methods. Prototype embedding has been shown to produce vectors for each object whose distance can be vastly different from their original one [14]. Using distance preserving embedding methods instead shows improvements of the median in several different types of datasets and distance functions [13]. These methods compute vectors x_i, x_j for objects o_i, o_j such that $\delta(o_i, o_j) \approx \delta_e(x_i, x_j)$, resulting in an embedding that more closely reflects the pairwise distances of the objects and thereby lead to a more accurate median computation. Combining the presented reconstruction methods and these distance preserving embedding methods may result in an even more accurate generalized median.

References

1. Bache, K., Lichman, M.: UCI Machine Learning Repository. School of Information and Computer Sciences, University of California, Irvine (2013)
2. Borzeshi, E.Z., Piccardi, M., Riesen, K., Bunke, H.: Discriminative prototype selection methods for graph embedding. *Pattern Recogn.* **46**(6), 1648–1657 (2013)
3. Cohen-Boulakia, S., Denise, A., Hamel, S.: Using medians to generate consensus rankings for biological data. In: Cushing, J.B., French, J., Bowers, S. (eds.) *SSDBM 2011*. LNCS, vol. 6809, pp. 73–90. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-22351-8_5](https://doi.org/10.1007/978-3-642-22351-8_5)
4. Ferrer, M., Karatzas, D., Valveny, E., Bardají, I., Bunke, H.: A generic framework for median graph computation based on a recursive embedding approach. *Comput. Vis. Image Underst.* **115**(7), 919–928 (2011)
5. Ferrer, M., Valveny, E., Serratoso, F., Riesen, K., Bunke, H.: Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recogn.* **43**(4), 1642–1655 (2010)
6. Franek, L., Jiang, X.: Ensemble clustering by means of clustering embedding in vector spaces. *Pattern Recogn.* **47**(2), 833–842 (2014)
7. Hakimi, S.: *Location Theory*. CRC Press, Boca Raton (2000)

8. de la Higuera, C., Casacuberta, F.: Topology of strings: median string is NP-complete. *Theoret. Comput. Sci.* **230**(1), 39–48 (2000)
9. Jiang, X., Munger, A., Bunke, H.: On median graphs: properties, algorithms, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(10), 1144–1151 (2001)
10. Jiang, X., Wentker, J., Ferrer, M.: Generalized median string computation by means of string embedding in vector spaces. *Pattern Recogn. Lett.* **33**(7), 842–852 (2012)
11. Krivanek, M., Moravek, J.: NP-hard problems in hierarchical-tree clustering. *Acta Informatica* **23**(3), 311–323 (1986)
12. Lundsteen, C., Phillip, J., Granum, E.: Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. *Clin. Genet.* **18**, 355–370 (1980)
13. Nienkotter, A., Jiang, X.: Distance-preserving vector space embedding for generalized median based consensus learning (2016, submitted for publication)
14. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. *Int. J. Pattern Recognit Artif Intell.* **23**(06), 1053–1081 (2009)
15. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. *J. ACM* **21**(1), 168–173 (1974)
16. Weiszfeld, E., Plastria, F.: On the point for which the sum of the distances to n given points is minimum. *Ann. Oper. Res.* **167**(1), 7–41 (2009)