ABCDepth: efficient algorithm
for Tukey depth

Milica Bogićević† and Milan Merklić‡

University of Belgrade, Faculty of Electrical Engineering, Bulevar Kralja
Aleksandra 73, 11120 Belgrade, Serbia

Abstract. We present a new algorithm for Tukey (halfspace) depth level sets
and its implementation. Given $d$-dimensional data set for any $d \geq 2$, the algorithm
is based on representation of level sets as intersections of balls in $\mathbb{R}^d$, and can be
easily adapted to related depths (Type D, Zuo and Serfling (Ann. Stat. 28 (2000),
461–482)). The algorithm complexity is $O(dn^2 + n^2 \log n)$ where $n$ is the data
set size. Examples with real and synthetic data show that the algorithm is much
faster than other implemented algorithms and that it can accept thousands of mul-
dimensional observations, while other algorithms are tested with two-dimensional
data or with a couple of hundreds multidimensional observations.

Keywords: Data centrality, Multivariate medians, Tukey depth, Algorithm, Com-
plexity computation.

1. Introduction

A basic statistical task is to simplify a large amount of data using some
values derived from the dataset as representative points. Among many ways
to choose representative points, a natural idea is to choose those that are
located in the center of data set. One way to define a center is to define
what is meant by deepness, and then to define the center as the set of
deepest points. Although this paper is about multivariate medians and
related notions, for completeness and understanding some ideas, we start
from the univariate case. Talking in terms of probability distributions, let $X$
be a random variable and let $\mu = \mu_X$ be the corresponding distribution, i.e.,
a probability measure on $(\mathbb{R}, \mathcal{B})$ so that $P(X \leq x) = \mu((-\infty, x])$. A median
of $X$ (or a median of $\mu_X$) is any number $m$ such that $P(X \leq m) \geq 1/2$ and
$P(X \geq m) \geq 1/2$. In terms of data set, this property means that to reach
any median point from outside of the data set, we have to pass at least 1/2
of data points, so this is the deepest point within the data set. Here we can
also find the depth of any point $x \in \mathbb{R}$ as

\begin{equation}
D(x, \mu) = \min\{P(X \leq x), P(X \geq x)\} = \min\{\mu((-\infty, x]), \mu([x, +\infty))\}
\end{equation}

The set of all median points \{Med $\mu$\} is a non-empty compact interval (can
be a singleton). It can be shown that (see [11], [12])

\begin{equation}
\{\text{Med } \mu\} = \bigcap_{J=[a,b]: \mu(J)>1/2} J,
\end{equation}

"antomripmuk@yahoo.com
†emerkle@etf.rs
and (2) can be taken for an alternative (equivalent) definition of univariate median set. In \( \mathbb{R}^d \) with \( d > 1 \), there are several different concepts of depth and medians (see for example [15], [17], [22]); the algorithm that we propose is applicable to so called type D depth functions (as in [22]) which can be obtained by generalizations of (1) to higher dimensions.

2. Depth functions based on families of convex sets

**Definition 2.1.** Let \( \mathcal{V} \) be a family of convex sets in \( \mathbb{R}^d \), \( d \geq 1 \), such that:

(i) \( \mathcal{V} \) is closed under translations and (ii) for every ball \( B \in \mathbb{R}^d \) there exists a set \( V \in \mathcal{V} \) such that \( B \subseteq V \). Let \( \mathcal{U} \) be the collection of complements of sets in \( \mathcal{V} \). For a given probability measure \( \mu \) on \( \mathbb{R}^d \), let us define

\[
D_{\mathcal{V}}(x; \mu) = \inf \{ \mu(U) \mid x \in U \in \mathcal{U} \} = 1 - \sup \{ \mu(V) \mid V \in \mathcal{V}, \ x \in V \}.
\]

The function \( x \mapsto D(x; \mu, \mathcal{V}) \) will be called a depth function based on the family \( \mathcal{V} \).

**Remark.** Definition 2.1 is a special case of Type D depth functions as defined in [22]. The conditions stated in [12] that provide desirable behavior of the depth function, are satisfied in this special case, with additional requirements that sets in \( \mathcal{V} \) are closed or compact.

**Example 2.1.**

1° Let \( \mathcal{V} \) be the family of all compact intervals \( [a, b] \subset \mathbb{R} \). As shown in [12], the depth function based on \( \mathcal{V} \) is the same as the one defined by (1).

2° With \( d = 2 \), consider the family \( \mathcal{V} \) of rectangles with sides parallel to coordinate axes. The corresponding depth function reaches its maximum \( D_{\text{max}} \geq 1/2 \) at coordinate-wise median. The same holds for \( d > 2 \), with "boxes" whose sides are parallel to coordinate hyper-planes.

3° For \( d > 1 \), let \( K \) be a closed convex cone in \( \mathbb{R}^d \), with vertex at origin, and suppose that there exists a closed hyperplane \( \pi \), such that \( \pi \cap K = \{0\} \) (that is, \( K \setminus \{0\} \) is a subset of one of open halfspaces determined by \( \pi \)). Define a relation \( \preceq \) by \( x \preceq y \iff y - x \in K \). Generalized intervals based on this partial order can be defined as

\[
[a, b] = \{ x \mid x - a \in K \wedge b - x \in K \} = (a + K) \cap (b - K).
\]

Now let us take \( \mathcal{V} \) to be a collection of all such intervals with finite endpoints and define the depth by (3). It can be shown ([12] Section 3) that the maximal depth is always \( \geq 1/2 \), and the median set can be found using formula (2) with generalized intervals.

4° Let us consider a family \( \mathcal{V} \) of all closed halfspaces in \( \mathbb{R}^2 \). Let \( X \) be a random point in \( \mathbb{R}^2 \) with \( \Pr(X = A) = \Pr(X = B) = \Pr(X = C) = 1/3 \), where \( ABC \) is an equilateral triangle. Here all points inside and on the border of the triangle \( ABC \) have the depth \( 1/3 \) and the depth of other points is equal to zero. Similar examples can be made for arbitrary dimension (see Example 4.1. in [12]).
From the above examples we see that

- A family $\mathcal{V}$ is not uniquely determined by the depth function; if we start with different collections $\mathcal{V}_1$ and $\mathcal{V}_2$, the corresponding depth functions based on them can be the same (see [12, Theorem 4.2] for a set of sufficient conditions).
- The maximal depth in $d > 1$ doesn’t need to be $1/2$ as in the scalar case. The following general result ([12, Theorem 4.1]) says that the maximal depth has to be $1/(d+1)$ or bigger:

**Theorem 2.1.** Let $\mathcal{V}$ be any non-empty family of compact convex subsets of $\mathbb{R}^d$ satisfying the conditions as in Definition 2.1. Then for any probability measure $\mu$ on $\mathbb{R}^d$ there exists a point $x \in \mathbb{R}^d$ such that $D_\mathcal{V}(x; \mu) \geq \frac{1}{d+1}$.

The set of points with maximal depth is called the center of distribution and denoted as $C(\mu, \mathcal{V})$. In general, one can observe level sets or depth regions of level $\alpha$ which are defined by

$$S_\alpha = S_\alpha(\mu, \mathcal{V}) := \{x \in \mathbb{R}^d \mid D(x; \mu, \mathcal{V}) \geq \alpha\}. \tag{4}$$

Clearly, if $\alpha_1 < \alpha_2$ then $S_{\alpha_1} \supseteq S_{\alpha_2}$ and $S_{\alpha} = \emptyset$ for $\alpha > \alpha_m$, where $\alpha_m$ is the maximal depth for given probability measure $\mu$.

The borders of depth level sets are called depth contours (in two dimensions) or depth surfaces in general. Let us note that the all statistical inference based on multidimensional depths is performed using level sets and contours (see [6, 7, 21, 14]), and that it is rarely necessary to find a depth of a particular point. On the other hand, in order to describe level sets and the center of distribution we do not need to calculate depth functions, as the next result shows (see Lemma 2.1 and Theorem 2.2. in [12]).

**Theorem 2.2.** Let $D(x; \mu, \mathcal{V})$ be defined for $x \in \mathbb{R}^d$ as in Definition 2.1. Then for any $\alpha \in (0, 1]$,

$$S_\alpha(\mu, \mathcal{V}) = \bigcap_{V \in \mathcal{V}, \mu(V) > 1-\alpha} V. \tag{5}$$

The center of a distribution is then the smallest non-empty level set; equivalently,

$$C(\mu, \mathcal{V}) = \bigcap_{\alpha: S_\alpha \neq \emptyset} S_\alpha(\mu, \mathcal{V}). \tag{6}$$

Since sets in $\mathcal{V}$ are convex, the level sets are also convex.

From (4) and (5) we can see that the depth function can be uniquely reconstructed starting from level sets.

**Corollary 2.1.** For given $\mu$ and $\mathcal{V}$, let $S_{\alpha}$, $\alpha \geq 0$ be defined as in (5), with $S_{\alpha} = \emptyset$ for $\alpha > 1$. Then the function $D : \mathbb{R}^d \mapsto [0, 1]$ defined by

$$D(x) = h \iff x \in S_{\alpha}, \alpha \leq h \quad \text{and} \quad x \notin S_{\alpha}, \alpha > h \tag{7}$$

is the unique depth function such that (4) holds.
The algorithm that we propose in this paper finds level sets, rather then the depth of particular points, based on the formula (5). The algorithm will be demonstrated in the case of half-space depth, which is described in the next section.

3. Half-space depth

The most popular choice among depth functions of Definition 2.1 is the one which is based on half-spaces, also called Tukey’s depth [20]. Here \( \mathcal{V} \) is the family of all open half-spaces, and the complements are closed half-spaces, so the usual definition of Tukey depth is obtained from (3) as

\[
D(x; \mu) = \inf \{ \mu(H) \mid x \in H \in \mathcal{H} \},
\]

where \( \mathcal{H} \) is the family of all closed halfspaces. In this section we consider only half-space depth, so we use the notation \( D(x, \mu) \) instead of \( D_{\mathcal{V}}(x, \mu) \).

As already noticed in Section 2, a depth function can be defined based on different families \( \mathcal{V} \). We say that families \( \mathcal{V}_1 \) and \( \mathcal{V}_2 \) are depth-equivalent if \( D_{\mathcal{V}_1}(x; \mu) = D_{\mathcal{V}_2}(x; \mu) \) for all \( x \in \mathbb{R}^d \) and all probability measures \( \mu \). Sufficient conditions for depth-equivalence are given in [12, Theorem 2.1], and it was shown there that in the case of half-space depth the following families are depth-equivalent: a) Family of all open halfspaces; b) all closed halfspaces; c) all convex sets; d) all compact convex sets; e) all balls.

For determining level sets we choose the last alternative - balls, and so we can define \( \mathcal{V} \) as a set of all balls (hyper-spheres) and level sets as

\[
S_\alpha(\mu, \mathcal{V}) = \bigcap_{B \in \mathcal{V}, \# \{ x_i : x_i \in B \} \geq n(1-\alpha)+1} B.
\]

In the setup with data sets, we have a set of \( n \) points \( \{ x_1, \ldots, x_n \} \) (with repetitions allowed) and we may use the counting measure defined as

\[
\mu(A) = \frac{\# \{ x_i : x_i \in A \}}{n}.
\]

Hence, the level sets in (9) for \( \alpha \in (0, 1] \) can be found as

\[
S_\alpha(\mu, \mathcal{V}) = \bigcap_{B \in \mathcal{V}, \# \{ x_i : x_i \in B \} \geq \lceil n(1-\alpha)+1 \rceil} B.
\]

In practical realization, in the step \( i \) we add one ball \( (B_i) \) that contains at least \( \lceil n(1-\alpha)+1 \rceil \) data points to intersection (11), so that in the \( k \)-the step we have \( S_{\alpha,k} = \bigcap_{i=1}^{k} B_i \). Clearly, \( S_{\alpha,k+1} \subseteq S_{\alpha,k} \). The process can end in two ways: (i) if for some \( K \) we do not have any data points in \( S_{\alpha,K} \) or (ii) if the number of data points in \( S_k \) remains the same for \( k \geq K \). In the case (i) we conclude that there are no data points \( x \) with \( D(x) \geq \alpha \). In the case (ii) the data points \( x \in S_{\alpha,K} \) have \( D(x) \geq \alpha \) as well as all points in their convex hull. In the similar way we can find the deepest point(s) among the points of given data set.
However, if we want to estimate level sets and especially the center of the underlying distribution from its sample, the described procedure may not be sufficient. As an example, consider a uniform distribution $\mu$ in the region bounded by circles $x^2 + y^2 = r_i^2$, $r_1 = 1$ and $r_2 = 2$. It is easy to see (from the original definition (8)) that the depth monotonically increases from 0 outside of the larger circle, to $1/2$ at the origin, which is the true median. With a sample from this distribution we will not have data points inside the inner circle.

In similar cases and whenever we have sparse data we can still visually identify depth regions and center, simply by adding artificial points to the data set. Let the data set contain points $x_1, \ldots, x_n$ and let $x_{n+1}, \ldots, x_N$ be points chosen from uniform distribution in some convex domain that contains the whole data set. Then we use a modification of the described procedure in such a way that we use augmented data set (all $N$ points) as a criterium for stopping (in cases (i) and (ii) above), but $n$ in formulas (10) and (11) is the cardinality of original data set.

In Section 4 we demonstrate the algorithm output for the sparse data set. The algorithm can be easily adapted to work with other convex sets, so the appropriate name is Algorithm based on Balls or other Convex sets to evaluate Depth, or ABCDepth. Although half-space depth is the only affine invariant among examples in Section 3, other options can be used in conjunction with data-driven coordinate system [3] and using the geometry of data sets.

4. ABCDepth: Implementation and The Output

According to ABCDepth, median and level sets can be obtained in three phases. In this section we describe each phase and its complexity. The detailed pseudocode is given in the appendix. For any further details about implementation authors will share the source code gladly.

**Phase 1:** In order to determine hypersphere that contain $[n(1-\alpha)+1]$ data points, we need $\frac{n(n-1)}{2}$ inter-distances. This triangular matrix of distances is stored as a list of lists, where $i$-th list $(i = 1, \ldots, n-1)$ contains distances $d_{i+1,j}$, $j = 1, \ldots, i$.

Each point is represented as a list of its coordinates, i.e. $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, so distance calculation between each two points needs to be performed taking $d$ steps.

Finally, the complexity of this phase is:

$$O(d\frac{n(n-1)}{2}) \sim O(dn^2).$$

**Phase 2:** Now we need to populate tuples. The tuples are represented as a map, where the key is a point, and the value is a list of (point, distance) pairs. In general, each entry in the map can be represented as $X_i \rightarrow list(X_j, d_{ij})$, where $j \in \{1, \ldots, n\} \setminus i$ and $d_{ij}$ is a
distance between $X_i$ and $X_j$ obtained from Phase 1. To populate this map, the algorithm takes $n$ steps for $n$ points, so the complexity of this step is $O(n)$.

Each value in the map, i.e. list of $(point, distance)$ pairs, is sorted by distance. To sort this list, we use the quicksort algorithm [9]. Its complexity is $O(n \log n)$.

At the end of Phase 2, we have populated and sorted tuples and the final complexity of this phase is:

$$O(n^2 \log n).$$

**Phase 3:** In Theorem 2.1 it is shown that $\alpha_{\text{max}} \geq \frac{1}{d+1}$.

For each value from the map ($O(n)$) populated in Phase 2, i.e. for each sorted list with $(point, distance)$ pairs, we take sublist that contains first $\lfloor n(1 - \alpha) + 1 \rfloor$ points ($O(1)$) and calculate intersections between those sublists ($O(n)$).

The algorithm repeats those steps iteratively for each obtained level set increasing $\alpha$ starting from $\frac{1}{d+1}$ in such manner that for each $S_k$ it holds: $\alpha_k = \alpha_{k-1} + \frac{1}{n}$, where $\alpha_1 = \frac{1}{d+1}$, until $S_k = S_{k+1}$ or until $S_{k+1} = \emptyset$. The data points $x \in S_k$ and all points in their convex hull are considered as a median and they have $D(x) \geq \alpha_k$.

Those steps result in an overall complexity of:

$$O(n^2).$$

ABCDepth produces a list of level sets where the last set in the list represents the median, i.e. points with the maximal depth. Namely, the sets in the list are ordered and $S_1 \supset S_2 \supset ... \supset S_k$, where $k$ is the total number of level sets, with $S_0$ representing the original data set.

The total algorithm complexity is:

$$O(dn^2) + O(n^2 \log n) + O(n^2) \sim O(dn^2 + n^2 \log n).$$

If the input set is sparse, ABCDepth optionally creates a new data set, i.e. multiset, $R' = R_1 \uplus R_2 = \{x_1 \ldots x_n\} \uplus \{x_{n+1} \ldots x_N\}$, where $R_1 = \{x_1 \ldots x_n\}$ is the original data set and $R_2 = \{x_{n+1} \ldots x_N\}$ is the data set that contains $N - n$ points generated from uniform distribution. The algorithm creates hyperspheres with the center in $x_i \in R', i = 1 \ldots N$ that contain $\lfloor n(1 - \alpha) + 1 \rfloor$ points from the original data set, $R_1$. The rest of the algorithm takes three phases we described above.

In Figure 1. we demonstrate an example when the data is rather sparse. The data set is the same data set as in [18]; it contains 23 four-dimensional observations in period from 1966 to 1967 that represent seasonally adjusted changes in auto thefts in New York city. For the sake of clarity, we take only two dimensions: percent changes in manpower, and seasonally adjusted
changes in auto thefts. The data is downloaded from [http://lib.stat.cmu.edu/DASL/Datafiles/nycrimedat.html](http://lib.stat.cmu.edu/DASL/Datafiles/nycrimedat.html). There are 4 level sets. The first level set, \( S_{\alpha,1} \), is bounded by the orange line and data points \( x \in S_{\alpha,1} \) have \( 0 \leq D(x) < 0.33 \). The second level set, \( S_{\alpha,2} \), is bounded by red line and its data points \( x \in S_{\alpha,2} \) have \( 0.33 \leq D(x) < 0.37 \). The third level set, \( S_{\alpha,3} \), is bounded by purple line and its data points \( x \in S_{\alpha,3} \) have \( 0.37 \leq D(x) < 0.42 \). The last level set, \( S_{\alpha,4} \) is considered as a median since its data points \( x \in S_{\alpha,4} \) has the maximum depth, \( D(x) \geq 0.42 \).

**Figure 1.** Level sets for the New York crime data set

On Figure 2. and Figure 3. are shown how \( n \) and \( d \) affect execution times with respect to the calculated complexity. Tests are performed on three-dimensional normal distribution, \( N_d(0, I) \). Measurements were taken for \( d \in \{2, ..., 10\} \), \( n \in \{40, 80, 160, 320, 640, 1280, 2560, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000\} \), and averaged on 10 runs.

Figure 4. shows level sets generated from 1000 points in dimension 2. The distribution is bivariate normal distribution and covariance matrix is identity matrix.

Figure 5. shows multivariate normal distribution constructed from 1000 points.

The initial version of ABCDepth algorithm was presented at [2]. All data generators we use in this paper in order to verify and plot the algorithm output was presented at [1] and it’s available as an open source project at [https://bitbucket.org/antomripmuk/generators](https://bitbucket.org/antomripmuk/generators).
5. Comparisons and Performances

During the last two decades, a couple of algorithms were proposed. One of the first was HALFMED [13]. Its complexity is also quadratic, $O(n^2 \log^2 n)$, but the algorithm is limited to bivariate data sets.

Another one, arguably the most popular, is DEEPLOC [18]. It calculates Tukey median in any dimension with complexity $O(kmn \log(n + kdn + md^3 + mdn))$, where $k$ is the number of steps taken by the program and $m$ is the number of directions, i.e. vectors constructed by the program. Measurements in DEEPLOC go up to 1000 points in dimension 5.

In [5] the approximation of the third method shown in [16] is represented. For one-dimensional and two-dimensional data Tukey depth is computed in $O(n)$ and $O(n \log n)$, respectively. For $d \geq 3$ becomes more complicated. Chan in [4] proposed an algorithm with complexity $O(n \log n)$ for $d < 3$ and $O(n^{d-1})$ for $d \geq 3$. It has lower computational complexity than ABCDepth for up to 3 dimensions, but it degrades quickly for higher dimensions. The algorithm from [4] has not been implemented so far.

Two algorithms are proposed in [10] for $p \geq 3$, with complexity $O(n^{d-1} \log n)$ and $O(\frac{n^{d-1}}{p} \log n)$. Their complexity is comparable to ABCDepth for up to dimension 3, but degrade for higher dimensions.

Recently, several algorithms for exact computation of Tukey depth was proposed in [8]. The first one reduces dimensionality to dimension one. The
Figure 3. The execution time grows linearly with dimensionality.

The overall complexity of the algorithm is $O(n^d)$. The second algorithm projects the data points into the 2-dimensional data space. The complexity of this algorithm is of order $O(n^{d-1} \log n)$. The third algorithm is recursive which results in the overall complexity of $O(n^{d-1} \log n)$.

Figure 6. shows IBM stock data distribution given in [10]: daily simple returns of IBM stock from January 1, 1970 to December 25, 2008. The data contains 9842 four-dimensional observations. For more information about the data see [19]. The data is downloaded from: http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts3/d-ibm3dx7008.txt.

Table 1. shows computation times for larger sizes of data sets and higher dimensions. The ABCDepth algorithm has been implemented in Java and tests are run using one kernel of Intel Core i7 (2.2 GHz) processor.

Although ABCDepth has demonstrated effectiveness of Tukey depth calculation, our plan is to implement a randomized algorithm that reduces the initial number of points, that should be considered as hyperspheres’ centers. Also, the effectiveness can be improved by dimensionality reduction, i.e. feature extraction.

An interesting idea would be to calculate the points’ distances as an approximation of Euclidian distance. If this is feasible, ABCDepth complexity would not depend on the number of dimensions.

The most obvious purpose of ABCDepth is classification of multidimensional data. Once the median and the level sets have been extracted from
Figure 4. The deepest point is at the origin.

|      | 4     | 5     | 6     | 7     | 8     | 9     | 10    |
|------|-------|-------|-------|-------|-------|-------|-------|
| 640  | 0.088 | 0.091 | 0.088 | 0.094 | 0.097 | 0.094 | 0.098 |
| 1280 | 0.273 | 0.286 | 0.286 | 0.301 | 0.315 | 0.310 | 0.309 |
| 2560 | 1.312 | 1.668 | 1.530 | 1.403 | 1.453 | 1.634 | 1.834 |
| 3000 | 1.846 | 1.921 | 1.931 | 2.261 | 2.367 | 2.166 | 2.231 |
| 3500 | 2.692 | 2.557 | 2.709 | 2.768 | 2.821 | 2.887 | 2.830 |
| 4000 | 3.320 | 3.560 | 3.662 | 3.734 | 3.817 | 3.854 | 3.817 |
| 4500 | 4.100 | 4.237 | 4.388 | 4.720 | 4.983 | 4.854 | 4.567 |
| 5000 | 4.986 | 5.355 | 5.304 | 5.264 | 5.451 | 5.550 | 5.613 |
| 5500 | 6.195 | 6.591 | 6.563 | 6.608 | 6.689 | 7.272 | 6.994 |
| 6000 | 7.348 | 7.507 | 8.943 | 7.880 | 8.111 | 8.242 | 8.266 |
| 6500 | 8.973 | 9.041 | 10.447| 9.591 | 9.788 | 10.164| 10.467|
| 7000 | 10.404| 10.811| 10.883| 11.364| 11.339| 11.285| 11.634|

Table 1. Computation times (in seconds)

the initial set, we can do an online classification of new data with same features. The data that belongs to same level set would be considered similar, based on minimal differences of their features.
Figure 5. 1000 points generated from 3D Gaussian distribution.

ACKNOWLEDGEMENT

We would like to show our gratitude to Anja Struyf, Univ. Instelling Antwerpen, for sharing the code and the data that was used in their papers that have the immense importance in this area.

APPENDIX A. PSEUDOCODE

The complete code takes three successive phases:

(1) Populate structure that stores distances between pairs of points.

calculate_distances(data):

    # init structure for storing distances: list of lists
    list_of_lists

    for i to data.length:  # O(n)
        ref_point = data[i];
Figure 6. IBM stock, data distribution. The median is represented as the smallest circle in the center.

```java
# init distance list for ith point list
for j = i to j < i:  # O((n-1)/2)
    point = data[j];
    list.add(ref_point.get_distance(point))  # O(d)

# add populated list to ith position in list_of_lists
list_of_lists.add(i, list);
```

(2) Populate map: key is a point, and the value is a list of (point, distance) pairs.

```java
populate_tuples(data):
    # init structure for storing tuples: map of lists
tuples_map
    for i to data.length:  # O(n)
```
ref_point = data[i];

# init tuples list for ith point
list

for j = 0 to j < data.length:  # O(n)
    # init tuple object
tuple = tuple.set(data[j]);
tuple = tuple.set_distance(list_of_lists(i, j));

    list.add(tuple);

    # sort tuples
    sort(tuples)  # O(nlogn)

    tuples_map.put(ref_point, list)

(3) Get the median and level sets.

get_median_level_sets(data, α):

    # init structure for storing median points: set
    set

    # calculate number of points
    # that spheres will contain.
    num = Math.floor(data.length * (1 - α) + 1)

    for i to tuples_map.size:  # O(n)
        sublist = tuples_map.get_val(i).sublist(0, num);

        set.retain(sublist);  # O(n)

    return set;

References

[1] M. Bogicevic and M. Merkle. Multivariate Medians and Halfspace Depth: Algorithms and Implementation. In Proc. 1st International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2014), Vrnjačka Banja, Serbia, volume 1, page 27, 2014.
[2] M. Bogicevic and M. Merkle. Data Centrality Computation: Implementation and Complexity Calculation. In Proc. 2nd International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2015), Srebrno Jezero, Serbia, volume 1, page 23, 2015.

[3] B. Chakraborty and P. Chaudhuri. On a transformation and re-transformation technique for constructing an affine equivariant multivariate median. Proc. Amer. Math. Soc., 124(8):2539–2547, 1996.

[4] T. M. Chan. An Optimal Randomized Algorithm for Maximum Tukey Depth. In Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 430–436. ACM, New York, 2004.

[5] Dan Chen, Pat Morin, and Uli Wagner. Absolute approximation of Tukey depth: Theory and experiments. Comput. Geom., 46:666–673, 2013.

[6] D. L. Donoho and M. Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. Ann. Statist., 20:1803–1827, 1992.

[7] S. Dutta, A. K. Ghosh, and P. Chaudhuri. Some intriguing properties of Tukey’s half-space depth. Bernoulli, 17:1420–1434, 2011.

[8] Rainer Dyckerhoff and Pavlo Mozharovskyi. Exact computation of the halfspace depth. Computational Statistics & Data Analysis, 98:19–30, 2016.

[9] C. A. R. Hoare. Algorithm 64: Quicksort. NSERC Research Grant, Comm. Acm., 321, 1961.

[10] Xiaohui Lui. Fast Implementation of the Tukey depth. arXiv, 2014.

[11] Milan Merkle. Jensen’s inequality for medians. Stat. Prob. Letters, 71:277–281, 2005.

[12] Milan Merkle. Jensen’s inequality for multivariate medians. J. Math. Anal. Appl., 370:258–269, 2010.

[13] P. J. Rousseeuw and I. Ruts. Constructing the Bivariate Tukey Median. Statist. Sinica, 8:827–839, 1998.

[14] P. J. Rousseeuw and I. Ruts. The depth function of a population distribution. Metrika, 49:213–244, 1999.

[15] Peter J. Rousseeuw and Mia Hubert. Statistical depth meets computational geometry: a short survey. arXiv, 2015.

[16] Peter J. Rousseeuw and Anja Struyf. Computing location depth and regression depth in higher dimension. Statistics and Computing, 8:193–203, 1998.

[17] C. G. Small. A survey of multidimensional medians. arXiv, 58:263–277, 1990.

[18] A. Struyf and P. J. Rousseeuw. High-dimensional computation of the deepest location. Comp. Statist. & Data Anal., 34:415–426, 2000.

[19] R.S. Tsay. Analysis of Financial Time Series, Third Edition. John Wiley & Sons., 2010.

[20] John Tukey. Mathematics and Picturing Data. In Proc. International Congress of Mathematicians, Vancouver 1974, volume 2, pages 523–531, 1975.

[21] Y. Zhou and R. Serfling. Multivariate spatial U-quantiles: A Bahadur-Kiefer representation, a Theil-Sen estimator for multiple regression, and a robust dispersion estimator. J. Statist. Plann. Inference, 138:1660–1678, 2008.

[22] Y. Zuo and R. Serfling. General notions of statistical depth function. Ann. Stat., 28:461–482, 2000.