*Journal of **AI** and **D**ata **M**ining*

A Grid-based Approach for Convexity Analysis of a Density-based Cluster

Anonymous Author 1*

Anonymous Affiliation Address 1
Anonymous Email 1
Anonymous Author 2, Anonymous Author 3
Anonymous Affiliation Address 2

Received ** *** ****; Revised ** *** ****; Accepted ** *** ****
*Corresponding author: Anonymous Email 1 (Anonymous Author 1)

Abstract

This paper presents a novel geometrical approach to investigate the convexity of a density-based cluster. Our approach is grid-based and we are about to calibrate the value space of the cluster. However, the cluster objects are coming from an infinite distribution, their number is finite, and thus, the regarding shape will not be sharp. Therefore, we establish the precision of the grid properly in a way that, the reliable approximate boundaries of the cluster are founded. After that, regarding the simple notion of convex sets and midpoint convexity, we investigate whether or not the density-based cluster is convex. Moreover, our experiments on synthetic datasets demonstrate the desirable performance of our method.

Key words. *Computational geometry, density-based cluster, convexity analysis, grid-based.*

1. Introduction

Convexity analysis of a set of points is an essential problem in computational geometry, as well as many other scientific fields. In the literature, there are some methods for calibrating the value space of the problem, like convex hull [3], voronoi diagram [2] and delaunay triangulation [6]. In convex hull, we are trying to find the smallest convex set that contains the cluster objects. For instance, for cluster $\mathcal{X}$ in two dimensions, the convex hull may be visualized as the shape enclosed by a rubber band stretched around $\mathcal{X}$. In voronoi diagram, we calibrate the value space by partitioning it into regions, based on distance to points in a specific subset of the plane. Also, a delaunay triangulation for a given set of points is a triangulation, such that no point will be inside the circumcircle of any triangle of related points [8]. But the problem is that the essence of such approaches is about a set of points, and none of them have been employed to analyze the convexity of a cluster. Therefore, we are about to adopt a new method from such geometrical approaches, which can discuss on convexity of a density-based cluster.

In this paper, we propose a novel grid-based approach, which is both easy-to-understand and easy-to-implement, for finding the extreme marginal points of a dense cluster. These points include not only the outer margin of the cluster, but even the inner margins of it will be detected. After gaining such points, which constitute the approximate frontiers of the cluster, we can assume the cluster as a multidimensional shape, whose boundaries are calibrated. Then by utilizing the concept of the midpoint convexity [4] on these boundary points, we can evaluate the convexity of the cluster.

The paper is organized as follows: In Section 2, we present the basic concepts required to understand the fundamentals of the proposed method. In Section 3, the detailed descriptions of the proposed approach along with the novel algorithms are provided. In Section 4, experimental analyses are reviewed. Finally, we conclude the paper in Section 5.

2. Basic Concepts

A number of definitions related to grid-based analysis of a cluster are reviewed. First of all, to find out that which points of the cluster are at its boundaries, we need to calibrate the space, which is covering the cluster, and it is like rasterizing a two-dimensional image, as then we know the exact position of every cluster point. Thus, we are utilizing a grid for this purpose. The major notations are represented in table 1.

Definition 1. (Grid Structure) Let the value space, which is about to be calibrated, consist of p variables, then a grid structure $\mathcal{G}$ is a partitioning of the data space, utilizing grid points, into finite number of non-overlapping hypercubic regions, called cells. The extrema of this grid in each dimension are taken as the minimum and maximum values of the related attribute.

Definition 2. (Grid Accuracy) The grid accuracy ε defines the size of every cell of $\mathcal{G}$ in each dimension.

Definition 3. (Neighboring Set of a Grid Point) For the arbitrary grid point $g = (g_1, \cdots, g_i, \cdots, g_p)$ in $\mathcal{G}$, its neighbors are defined as those grid points with the exact dis-

Table 1 . Major Notations

Notation	Description
$\mathcal{G}$	The grid structure covering the value space
ε	The grid accuracy
t	Number of grid points in $\mathcal{G}$
η	Random sampling rate for grid points
$\mathcal{G}_s$	Set of sampled grid points
$\mathbb{N}(g)$	The neighboring set of grid point g
$[X]_{n\times p}$	Input cluster X with n objects and p attributes
$\mathcal{N}_\varepsilon(x)$	The ε neighborhood of point x
$D(x,y)$	Euclidean distance between points x and y
$M(x,y)$	Midpoint of points x and y
ω	Convexity status of a cluster after analysis
ψ	Candidate point for proving the non-convexity of a cluster

tance of ε from it. It is like as these neighboring points are lying on a hypersphere, with g as its centroid and ε as its radius. Hence, w.r.t. the grid accuracy, the neighboring grid points of g in the jth dimension will be $(g_1,\cdots,g_i-\varepsilon,\cdots,g_p)$ and $(g_1,\cdots,g_i+\varepsilon,\cdots,g_p)$. That is:

$$\mathbb{N}(g)=\bigcup_{i=1}^{p}(g_1,\cdots,g_i\pm\varepsilon,\cdots,g_p)$$

Therefore, one can state that for every grid point in $\mathcal{G}$, there are $2p$ neighbors[1].

Definition 4. (ε Neighborhood of a Point) For arbitrary point x in euclidean space, its ε neighborhood is a set which contains all arbitrary objects with a distance less than or equal to ε. That is:

$$\mathcal{N}_\varepsilon(x)=\{y\mid D(x,y)\le\varepsilon\},y\in\mathcal{N}_\varepsilon(x)\iff x\in\mathcal{N}_\varepsilon(y)$$

where $D(x,y)$ denotes the euclidean distance between objects x and y. Moreover, ε vicinity is a symmetric concept.

3. Proposed Approach

Our proposed approach consists of three major phases. At the first phase, we find those grid points which do not fall at the coverage area by the cluster shape. At the second phase, we utilize such points to detect cluster points which are located at its inner and outer edges. Finally, at the third phase, we evaluate the midpoint convexity on these edging points.

Remark 1. According to the fact that, number of grid points t, increases exponentially w.r.t. the number of dimensions p, hence a situation might happen in which, the amount of grid points is so huge that cannot fit into memory, and also, the consequent computational complexity would be intolerable too. Therefore, in a preprocessing step, one can resort to dimensionality reduction approaches, as in them, the pairwise euclidean distances between data points are approximately preserved. Popular methods are PCA [10] and Random Projections (RP) [1, 9]. The main difference between these methods is that PCA is computationally more expensive than RP, but in reverse, its accuracy is more than RP in most cases. Moreover, PCA is more sensitive to the choice of the number of reduced dimensions, while the accuracy for RP increases normally with the number of dimensions, as long as it is desirable for lower dimensions too [5].

Remark 2. Sometimes, even by reducing the dimensions of the value space, the amount of grid points is still far high, that would take so much time to be processed. Hence, by losing some precision, in the same preprocessing step, one can conduct a random sampling on grid points, with a reasonable rate, and still expect remarkable results out of the experiments[2].

Remark 3. For the grid accuracy, we have to define it in an appropriate manner, which would lead to precise detection results. Here, we prefer to use the neighborhood parameter epsilon of DBSCAN algorithm [7], as the grid accuracy. We consider the optimal value for epsilon, as by which, DBSCAN will report a unique density-based cluster without any noise, as its output. The reason is described as follows. In truth, we can divide the value space containing the cluster into two distinct partitions. One is the space, which is covered by the cluster shape, if we consider it as a distribution with an infinite number of points. The other partition will be the space not including any points of that distribution.

Now, if we intend to establish the grid in a way that the frontiers of the cluster will be defined properly, we should set the grid accuracy in a manner which by that, every grid point which is in the coverage region of the cluster, will be in the ε neighborhood of at least one cluster point.

If the grid accuracy is set too low, then there will be some grid points which cannot take place in the ε neighborhood of any cluster point, while they are in the covered region by the cluster. In reverse, if it is established too high, then the frontiers of the cluster will not be discovered with desirable precision.

Therefore, we define the accuracy value of the grid equal to the epsilon parameter of DBSCAN, as it is good enough to contain all the grid points in the coverage area by the cluster, and also, to establish the reliable boundaries of the cluster[3].

The framework of proposed approach is presented in Algorithm 1, which consists of a preliminary phase and three other major phases including: 0) Initializing the grid; 1) Detecting non-neighboring grid points; 2) Finding marginal cluster points; 3) Analyzing midpoint convexity. All of these phases will be described in details in following subsections.

3.0. Initializing the grid

At this early stage, we establish the grid structure which is covering the value space of the cluster. For this matter we should divide each dimension into smaller pieces equal to ε in size. But as for some extreme grid points which might

[1] For grid points at the extrema of the grid, the number of neighbors would be less.

[2] Sometimes, in a case that the number of cluster objects is very high, we can carry out a random sampling on data points too, to lower the computational complexity. But this sampling rate should not be so low by which, the structure of the cluster would become so sparse and distorted, and thus, not reliable for being investigated for convexity.

[3] With increase in the density of the distribution, the value of ε will decrease, and the final precision will increase.

Algorithm 1: $[\omega, \psi]$ = GridConvxAnals$(\mathcal{X}, \varepsilon, \eta)$

Input : $\mathcal{X}$ - Input cluster; ε - Grid accuracy; η - Random sampling rate for grid points

Output: ω - Convexity status; ψ - Candidate point for non-convexity

Phase 0 — Initializing the grid:

Step 1. Add a distance of 2ε to the extrema of the value space in each dimension, and initialize the grid structure $\mathcal{G}$, regarding the grid accuracy ε

Step 2. Create ηt number of grid points totally at random, w.r.t. $\mathcal{G}$, and assign them to $\mathcal{G}_s$

Phase 1 — Detecting non-neighboring grid points:

Step 3. For every grid point in $\mathcal{G}_s$ evaluate whether or not it falls in the ε neighborhood of any cluster point, and separate the non-neighboring grid points as a distinct set, all w.r.t. Algorithm 2

Phase 2 — Finding marginal cluster points:

Step 4. According to Algorithm 3, regarding the set of non-neighboring grid points, find those neighbors of these points, which are at the ε neighborhood of at least on cluster point. Consider the nearest cluster point as one of the frontier points of the cluster

Phase 3 — Analyzing midpoint convexity:

Step 5. With respect to Algorithm 4, for every distinct pair of frontier points of the cluster, find the related midpoint, and analyze whether this midpoint falls in the ε neighborhood of any cluster point. If it is not so, then announce that the cluster is non-convex, and provide that midpoint as a candidate point for proving non-convexity of the cluster

be at the ε neighborhood of a cluster object, and thus, the related marginal cluster points will not be capable of being found in our process, therefore, we add a distance of 2ε to the extrema of each dimension of the value space. Hence, even at the extreme regions of a cluster in any dimension, there will be at least one grid point which is not covered by the cluster shape.

After creating the grid vectors, by simple randomness, we choose a value in each vector and thus, create an arbitrary grid object. As we are going to work with a portion of total grid points, we conduct the process multiple times and create the set of sampled grid points. In the following, this set will represent the whole grid with some precision proportional to the random sampling rate η for grid structure.

3.1. Detecting Non-Neighboring Grid Points

After attaining the sampled grid, we afford to find those grid objects which do not fall in the ε neighborhood of any cluster object. In other words, such points are not covered by the cluster shape. The reason that at the first step, we are trying to locate these points is that by utilizing them, we can find those grid neighbors of them, which are at the boundaries of the cluster. Algorithm 2 demonstrates the required steps for finding non-neighboring grid points.

3.2. Finding Marginal Cluster Points

By gaining the non-neighboring grid points, as it was mentioned earlier, we afford to find those neighbors of

Algorithm 2: $[\mathcal{T}]$ = DetcNonNghbGridPnts$(\mathcal{X}, \mathcal{G}_s, \varepsilon)$

Input : $\mathcal{X}$ - Input cluster; $\mathcal{G}_s$ - Set of sampled grid points; ε - Grid accuracy

Output: $\mathcal{T}$ - Set of non-neighboring grid points

```
T ← Φ
foreach g ∈ G_s do
    ζ ← 0
    foreach x ∈ X do
        if g ∈ N_ε(x) then
            ζ ← 1
            break
        end
    end
    if ζ ≡ 0 then
        T ← T ∪ g
    end
end
```

such grid objects, which fall at the ε neighborhood of at least one cluster object[4]. By finding that closest cluster object, we mark it as one of the bordering points of the cluster shape. Algorithm 3, shows the required steps.

Algorithm 3: $[\mathcal{V}]$ = DetcMargClstPnts$(\mathcal{X}, \mathcal{T}, \varepsilon)$

Input : $\mathcal{X}$ - Input cluster; $\mathcal{T}$ - Set of non-neighboring grid points; ε - Grid accuracy

Output: $\mathcal{V}$ - Set of marginal cluster points

```
U ← Φ
V ← Φ
foreach h ∈ T do
    U ← U ∪ N(h)
end
U ← U \ T
foreach i ∈ U do
    foreach x ∈ X do
        if i ∈ N_ε(x) then
            V ← V ∪ x
            break
        end
    end
end
```

3.3. Analyzing Midpoint Convexity

After building the approximate boundaries of the cluster structure, it is time to analyze the notion of midpoint convexity on the marginal cluster points. Therefore, we evaluate every distinct pair of such points, whether or not their midpoint falls in the ε neighborhood of at least one cluster object. If there is at least one pair of frontier points, which their midpoint is not covered by the cluster shape, hence, the cluster will be reported as non-convex. Otherwise, it would be convex with the precision of ε. Algorithm 4, illustrates the needed actions to evaluate midpoint convexity on frontier objects of the cluster.

[4] If the whole grid is utilized, all of the neighbors of non-neighboring grid points, excluding the non-neighboring objects themselves, will be the same marginal grid points. And we just need to find the closest cluster object to each of them.

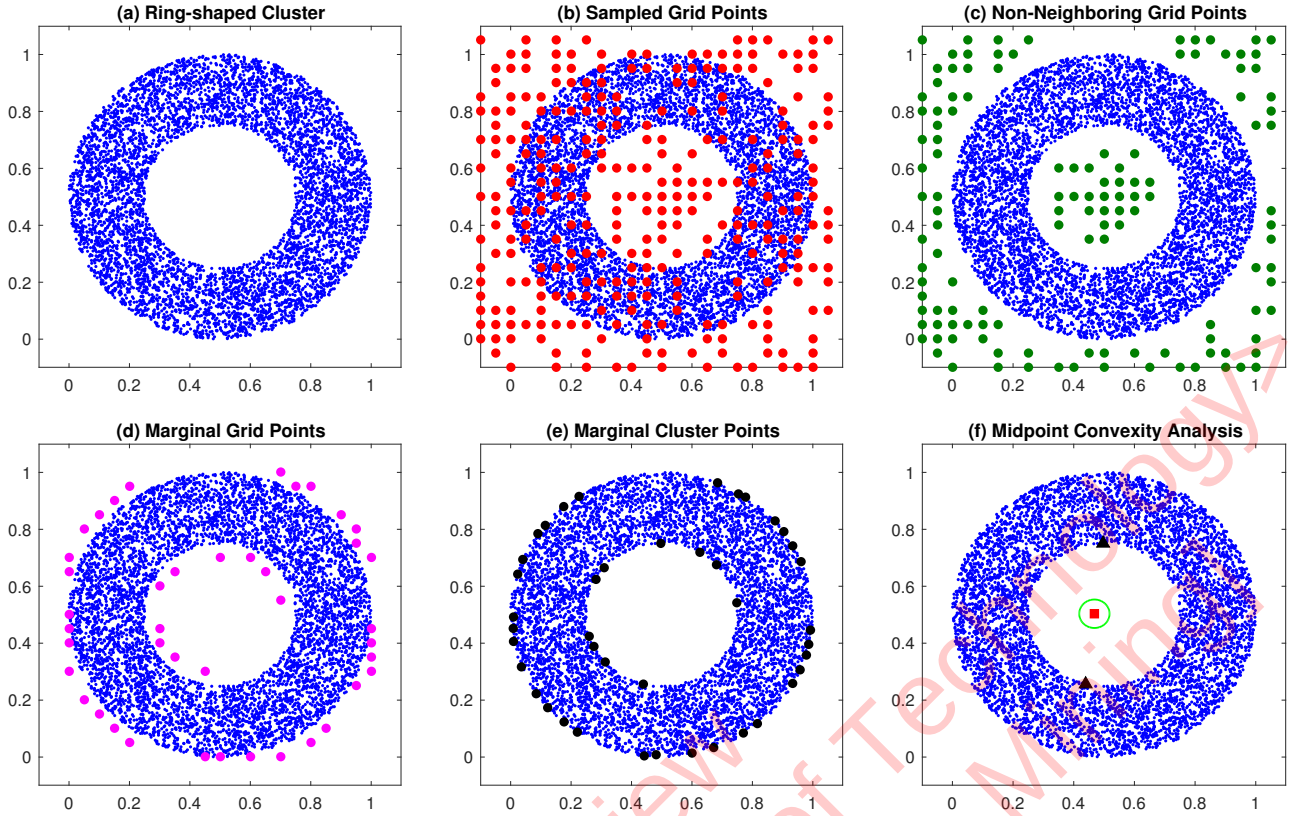

Figure 1 . Convexity analysis of a 2D ring-shaped cluster

Algorithm 4: $[\omega, \psi]$ = MidPntCnvxAnls$(\mathcal{X}, \mathcal{V}, \varepsilon)$

Input : $\mathcal{X}$ - Input cluster; $\mathcal{V}$ - Set of marginal cluster points; ε - Grid accuracy

Output: ω - Convexity status; ψ - Candidate point for non-convexity

$\omega \leftarrow 1$
$\psi \leftarrow \phi$
foreach $(j,k) \mid j,k \in \mathcal{V}$ *and* $j \neq k$ **do**
  $\gamma \leftarrow M(j,k)$
  $\zeta \leftarrow 0$
  foreach $x \in \mathcal{X}$ **do**
    if $\gamma \in \mathcal{N}_{\varepsilon}(x)$ **then**
      $\zeta \leftarrow 1$
      break
    end
  end
  if $\zeta \equiv 0$ **then**
    $\omega \leftarrow 0$
    $\psi \leftarrow \gamma$
    break
  end
end

4. Experiments

In this section, we conduct an efficacy test, to show that our algorithm is capable of detecting the reliable rough margins of a non-convex dense cluster, and proves its non-convexity in a geometrical manner. Moreover, a test is carried out to illustrate the serious dependence of the proposed method on the grid accuracy. Another test is presented to show that even by low rates of random sampling, one can still expect significant detection results out of our approach. All implementations are carried out with MATLAB 9, and run on a laptop with Intel Core i5 processor (clocked at 2.5 GHz) and 6 G memory.

4.1. Efficacy Test

While there is not any real-life benchmark data which claims on the convexity of its contained clusters, thus, we run evaluations on a synthetic two-dimensional dataset, with uniform distribution, including only one non-convex cluster in the shape of a ring.

Figure 1a demonstrates the ring-shaped 2D cluster, denoted as blue dots, with $\varepsilon = 0.05$, as the grid accuracy, as by which, the DBSCAN algorithm will report the related cluster, as a dense shape and without any noise. Figure 1b demonstrates the cluster objects along with the sampled grid points shown with red circles, created through the phase of initializing the grid structure, with random sampling rate $\eta = 0.5$. Figure 1c represents the non-neighboring grid points with green circles, and figure 1d illustrates the marginal grid points in magenta circles, which are at the neighborhood of at least one non-neighboring grid object.

Moreover, figure 1e demonstrates the frontier cluster objects, denoted as black circles, obtained with the precision of ε, and as it is clear, both outer and inner frontiers are detected. Finally, figure 1f shows a pair of marginal cluster objects denoted as black triangles, and their midpoint denoted as a red square, with its ε vicinity denoted as a green

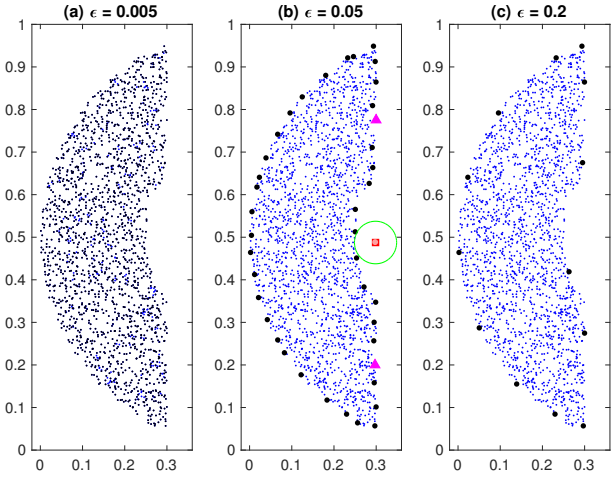

Figure 2 . Effect of grid accuracy on detection result

circle. It is crystal clear that the midpoint is not in the ε neighborhood of any cluster object. Therefore, the dense ring-shaped cluster is reported as a non-convex shape.

4.2. Test on Grid Accuracy

At this part, we demonstrate that if the grid accuracy is not established in a proper manner, then not only the non-convexity might not be discovered, but even a convex-shaped cluster could be declared as non-convex. As it was explained earlier, if the grid accuracy is set too low, then there will be some grid points, erroneously reported as non-neighboring objects. And if it is set too high, the consequent boundaries will not be reported appropriately. Figure 2 illustrates a crescent-like density-based cluster, denoted as blue dots, in three different conditions made by various values for grid accuracy.

Figure 2a shows the crescent along with the marginal cluster points, denoted as black dots, obtained through $\varepsilon = 0.005$. As it is clear, not only the boundaries of the crescent are not detected correctly, but even there are numerous cluster points mistaken as frontier objects. Figure 2b demonstrates the best result achieved through $\varepsilon = 0.05$. Two arbitrary marginal cluster objects and their midpoint, along with its ε vicinity, are represented with magenta triangles and a red square and a green circle, respectively. As it is apparent, the midpoint is out of the contained area by the crescent, and hence, convexity will be reported. Figure 2c shows the crescent along with the boundary objects obtained through $\varepsilon = 0.2$. As it is obvious, the related boundaries are not precisely detected, thus, there is not any distinct pair of marginal cluster points with a midpoint out of the coverage region by the cluster, w.r.t. the elected high value for grid accuracy.

4.3. Test on Sampling Rate for the Grid

However, there is no guarantee that with very low rates of random sampling for the grid, one can still anticipate remarkable detection results out of our proposed approach, we illustrate here with two examples run with different very low values of η, that it could be possible to discover

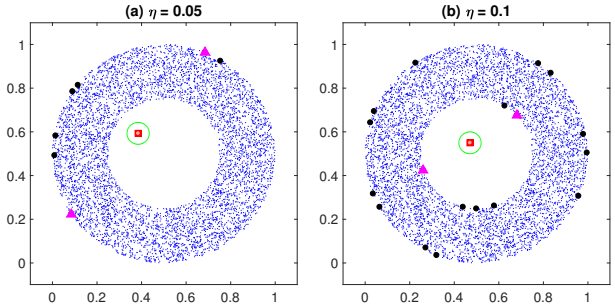

Figure 3 . Effect of random sampling rate for the grid, on detection result

the non-convexity, even with very small samples of the grid.

As the random sampling rate decreases, the chance at which both inner and outer frontier objects could be detected correctly, reduces monotonically. Figure 3 demonstrates the same ring from efficacy test, with the same graphical representations as in figure 2. Figure 3a shows the ring and the related marginal cluster points obtained through $\eta = 0.05$. It is clear to see that not any points of the inner margin are detected. But as the position of the inner hole is at the exact center of the ring, there is a pair of outer marginal cluster points that their midpoint falls in the hole, far from the coverage region by the ring. Hence, the ring is declared as a non-convex density-based cluster. Figure 3b demonstrates the result achieved out of $\eta = 0.1$. However, both inner and outer boundaries of the shape are detected with a very low precision, the non-convexity of the ring is elicited successfully. Therefore, one can state that for better and more accurate detection outcomes, we need to utilize large enough samples of the grid, otherwise the outcome might be unreliable.

5. Conclusion

In this paper, we just provided a new plain approach, risen from the field of computational geometry, to examine the convexity of a density-based cluster. For this matter, first of all we initialized a grid and tried to locate those grid objects, which are not covered by the cluster shape. By utilizing these objects, we afforded to find marginal cluster objects, and built the approximate inner and outer frontiers of the cluster upon them. Finally, by employing the concept of midpoint convexity on these boundary points, we discovered whether the cluster shape is convex.

Acknowledgment

This research has been funded by the Engineering Department of Bozorgmehr University of Qaenat, under grant number 39158.

References

[1] Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the twentieth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 274–281).

[2] Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, *23*(3), 345–405.

[3] Berg, M. d., Cheong, O., Kreveld, M. v., & Overmars, M. (2008). *Computational geometry: algorithms and applications*. Springer-Verlag TELOS.

[4] Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[5] Deegalla, S., & Bostrom, H. (2006). Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *2006 5th international conference on machine learning and applications (icmla'06)* (pp. 245–250).

[6] Delaunay, B., et al. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, *7*(793-800), 1–2.

[7] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, pp. 226–231).

[8] Fisher, J. (2004). Visualizing the connection among convex hull, voronoi diagram and delaunay triangulation. In *37th midwest instruction and computing symposium.*

[9] Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, *26*(189-206), 1.

[10] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.