A New Concave Hull Algorithm and Concaveness Measure for *n*-dimensional Datasets*

JIN-SEO PARK AND SE-JONG OH
Department of Nanobiomedical Science and and WCU Research Center of Nanobiomedical Science
Dankook University
Cheonan, 330-714, South Korea

Convex and concave hulls are useful concepts for a wide variety of application areas, such as pattern recognition, image processing, statistics, and classification tasks. Concave hull performs better than convex hull, but it is difficult to formulate and few algorithms are suggested. Especially, an *n*-dimensional concave hull is more difficult than a 2- or 3- dimensional one. In this paper, we propose a new concave hull algorithm for *n*-dimensional datasets. It is simple but creative. We show its application to dataset analysis. We also suggest a concaveness measure and a graph that captures geometric shape of an *n*-dimensional dataset. Proposed concave hull algorithm and concaveness measure/graph are implemented using java, and are posted to http://user.dankook.ac.kr/~bitl/dkuCH.

Keywords: convex hull, concave hull, classification, dataset analysis, time complexity

1. INTRODUCTION

The term 'convex hull' indicates the boundary of the minimal convex set containing a given non-empty finite set of points in the plane (or *n*-dimensional space), as shown in Fig. 1. A convex hull has been used in practical applications, in pattern recognition, image processing, statistics, and so on [16-22]. Many algorithms are proposed to find a convex hull in the 2-dimensional space (plane), and a few specialized algorithms for 3-dimensional space are developed [8-15,26]. Recently, researchers have applied the convex hull to classification tasks. Jianjun *et al.* [7] proposed a new classification method based on the convex hull. In this, convex hulls are produced for each class of known objects and the distance between each convex hull and unknown object is then calculated, with the nearest convex hull being the predicted class for the unknown object.

From the simple example shown in Fig. 2, we can know that the convex hull does not fully reflect the geometrical characteristics of a dataset and that the 'concave hull' is a better choice for geometrical evaluation. The concave hull approach is a more advanced approach used to capture the exact shape of the surface of a dataset; however, producing the set of concave hull is difficult. Little work has focused on concave hull algorithms. Galton and Duckham [24] suggested 'Swing Arm' algorithm based on gift-wrapping algorithm. The Swing Arm Algorithm may produce separated concave hulls instead of single one. Adriano and Yasmina [26] suggested a concave hull algorithm based on the

* This work was supported by grant No. R31-2008-000- 10069-0 from the World Class University (WCU) project of the Ministry of Education, Science & Technology (MEST) and the Korea Science and Engineering

k-nearest neighbors approach. The LOCAL project website (http://get.dsi.uminho.pt/local/) supports an online version of the concave hull algorithm. Duckham *et al.* [25] suggested 'χ-shape' algorithm based on Delaunay triangle. Those algorithms are for 2-dimensional datasets; extending to 3- or higher dimensions is difficult or impossible whereas a lot of datasets for classification have three or more dimensions. If we want to apply a concave hull for the areas such as bioinformatics, we need to develop a multi-dimensional concave hull algorithm.

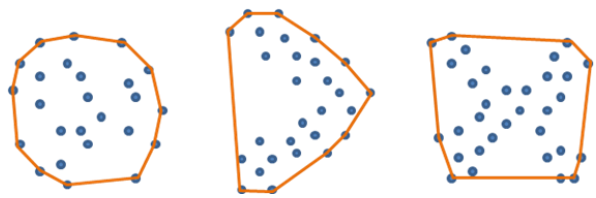
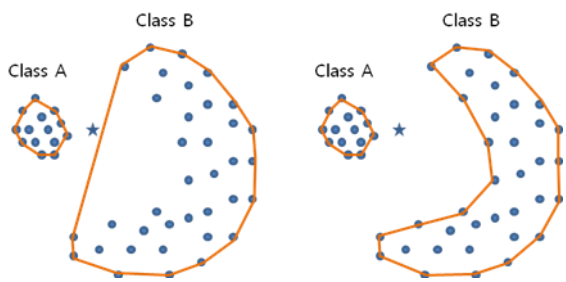


Fig. 1. Example of 2-dimensional convex hull



(a) convex hull (b) concave hull

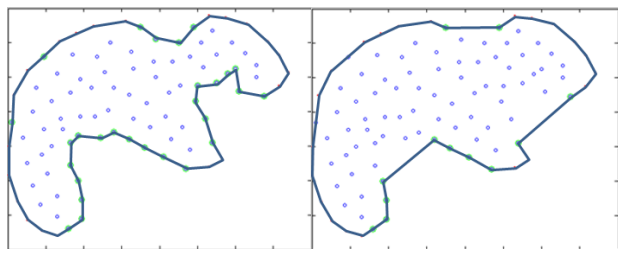
★ is belong to class B ★ is belong to class A

Fig. 2. Classification using convex hull and concave hull

2. A NEW CONCAVE HULL ALGORITHM

2.1 Smoothness of concave hull

Our strategy to produce a set of concave hulls was to identify a convex hull using a known algorithm, and 'dig' it to produce a concave hull with an appropriate depth. The digging level was determined by the threshold value *N*. Using a smaller *N* induces a sharper surface (*See* Fig. 3). If *N* has a sufficiently large value, our algorithm does not dig, and will return a convex hull, instead of a concave hull. Threshold *N* determines the smoothness of a concave hull, and the optimal value of *N* depends on applications and characteristics of their datasets. Some applications need a sharp concave hull and others need a rounded concave hull. Therefore, planners of concave hull applications must select an approximate value of *N*.



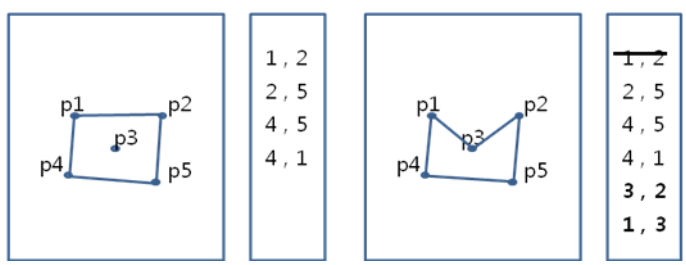
Threshold = 1.5 Threshold = 3.0

Fig. 3. Smoothness of concave hull by threshold.

The following sections describe a new concave hull algorithm, and concaveness measure as an application of the concave hull.

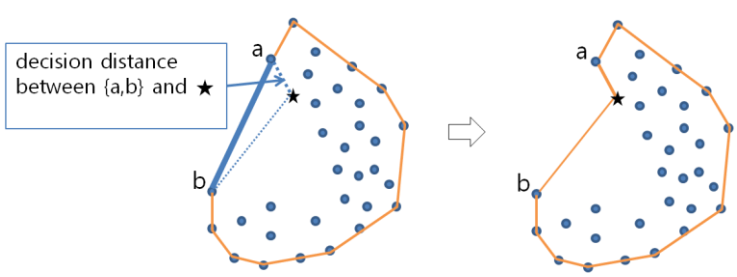
2.2 2-dimensional concave hull algorithm

For easy understanding, we introduce 2-dimensional algorithm, and extend it to 3- or higher dimensional algorithm. The proposed concave hull algorithm is composed of four steps. First, a set of convex hull edges is selected (*see* Fig. 4 (a)) and the threshold value *N* is chosen. Second, the nearest inner points from the convex hull edge are identified, after which the shortest distance between the nearest inner point and the edge's points is identified. This distance is called as a 'decision distance' (*see* Fig. 5 (a)). Third, a deci-



(a) *convexList* before digging (b) *concaveList* after digging

Fig. 4. Example of ConcaveList



(a) before digging (b) after digging

Fig. 5. Example of decision distance and digging

sion to dig or not is made by comparing N with the decision distance. If (length of edge)/(decision distance) > N, then we execute digging process (*see* Fig. 5 (b)). Fourth, the second and third procedures are repeated until there is no inner point for digging. The final output of this algorithm is a set of edges that forms a concave hull for the given dataset (*see* Fig. 4 (b)). For clarity we define some terms that are used in our algorithm.

Definition 1 *Input data* $G = \{p_1, p_2, p_3, .., p_k, .., p_n\}$ *is a set of points that belong to given dataset where n is a number of elements.*

If the given dataset is 2-dimensional, each point is expressed by a coordinate of (x, y) in a plane.

Definition 2 *ConvexList*(G) = $\{(c_{11},c_{12}), (c_{21},c_{22}), .., (c_{m1},c_{m2})\}$ *is a set of convex* edges where (c_{i1},c_{i2}) *is an edge consisting of two index numbers corresponding to points in G.*

In the example shown in Fig. 6 (a), *ConvexList*(G) = {(1,2), (2,5), (4,5), (4,1)}, and edges (1,2) = $\overline{p_1p_2}$, (2,5) = $\overline{p_2p_5}$, (4,5) = $\overline{p_4p_5}$, and (4,1) = $\overline{p_4p_1}$.

Definition 3 *Assuming function* $D(x, y)$ *returns a distance between point x and y, the decision distance* $DD(p, q)$ *between point p and set of points* $q=\{q_1, q_2, q_3, .., q_n\}$ *is defined by*:

$$DD(p ,q) = \min\{D(p, q_1), D(p, q_2), , D(p, q_3),.. , D(p, q_n)\} \quad (1)$$

Definition 4 $DE(p, (e_1,e_2))$ *is the distance from a point p to an edge* (e_1,e_2)*, and is defined by*:

$$DE(p, (e_1,e_2)) = \min\{ e_k \in (e_1,e_2) \mid D(p, e_k)\} \quad (2)$$

Definition 5 $DT(p, (t_1,t_2,t_3))$ *is the distance from a point p to a triangle* (t_1,t_2,t_3) *and is defined by*:

$$DT(p, (t_1,t_2,t_3)) = \min\{ t_k \in (t_1,t_2,t_3) \mid D(p, t_k)\} \quad (3)$$

The proposed concave hull algorithm is described as follows:

Algorithm 1 Concave hull algorithm for 2-dimensional dataset

```
/**
   Input: G                  // target dataset
   Output: ConcaveList       // set of edges that forms 2 dimensional concave hull
**/
```

```
/* 1. Preprocessing */
    generate ConvexList(G);                          // using a known algorithm
    choose the threshold N;
    copy ConvexList(G) to ConcaveList;

/* 2. Digging convex */
    for i=1 to the end of the ConcaveList
    {
        find the nearest inner point p_k ∈ G from the edge (c_i1,c_i2);
        // p_k should not closer to neighbor edges of (c_i1,c_i2) than (c_i1,c_i2)

        calculate eh=D(c_i1,c_i2);                   // the length of the edge
        calculate dd=DD(p_k, {c_i1,c_i2});

        if (eh/dd) > N                               // digging process
        {
            insert new edges (c_i1,k) and (c_i2,k) into the tail of ConcaveList;
            delete edge (c_i1,c_i2) from the ConcaveList;

        }
    }

    Return ConcaveList;
```

As mentioned above, the N threshold influences the smoothness of the concave hull. From the several experiments, we found that the valid range of N falls in [0, 5]. In most of cases, if $N > 5$, digging process is not working, and concave hull is same as convex hull.

The proposed concave hull algorithm works well in producing a set of concave hull data. Fig. 6 shows an example of forming a concave hull from a convex hull following the proposed algorithm. In the case of Fig. 6, the algorithm iterates 24 times through the digging loop. Each loop digs into the convex hull more and more deeply, and completes the concave hull.

2.3 Extending to 3-dimensional concave hull algorithm

In general, the shape of the concave hull depends on the dataset dimensions. If a dataset is *2*-dimensional, then the concave hull looks like a polygon chain. The foundational component of a concave hull is the 'edge', which is composed of two points. In the case of a *3*-dimensional dataset, the concave hull is solid and its foundational component is a plane of a 'triangle' that is composed of three points. *ConvexList* or *ConcaveList* of an *n*-dimensional dataset has *n* points for each component (*See* Fig. 7).

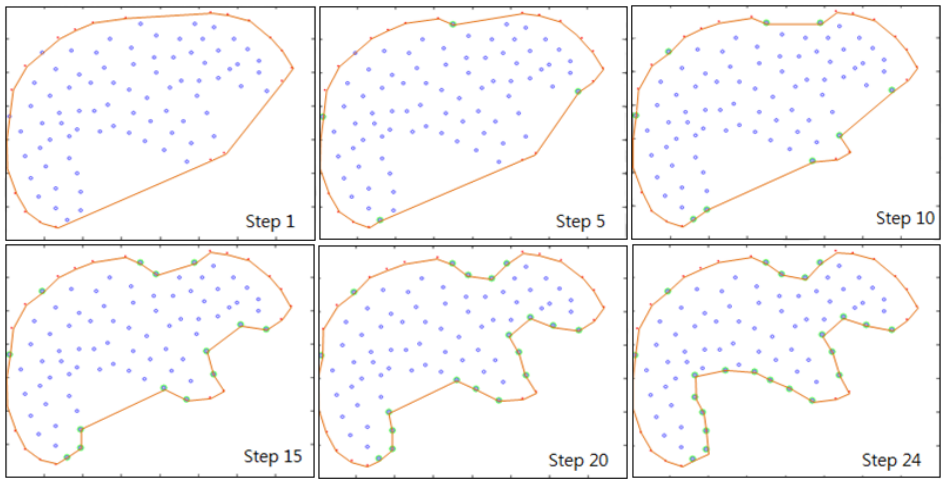


Fig. 6. Series of digging procedure

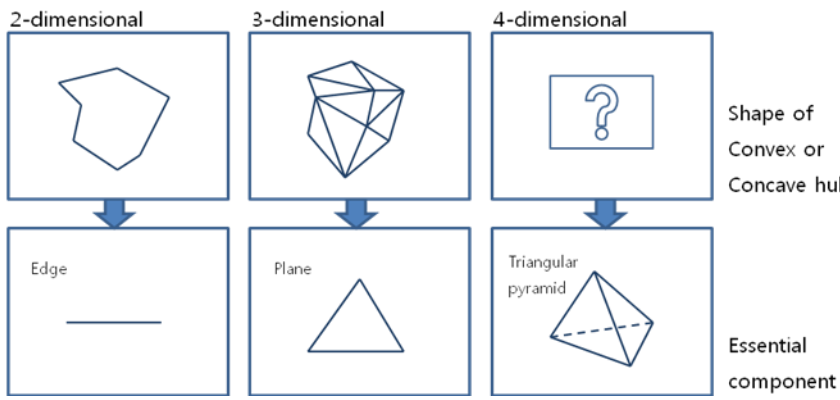


Fig. 7. Essential component of concave hull according to its dimension.

In the 3-dimensional dataset, each point of input data G is expressed by a coordinate of (x, y, z) in a solid. Each element of *ConvexList*(G) is expressed by (c_{i1}, c_{i2}, c_{i3}). The concave hull algorithm for 3-dimensional dataset is as follows:

Algorithm 2 Concave hull algorithm for 3-dimensional dataset

```
/**
   Input: G                  // target dataset
   Output: ConcaveList       // set of triangle that forms 3 dimensional concave hull
**/
```

```
/* 1. Preprocessing */
    generate ConvexList(G)                          // using a known algorithm
    choose the threshold N;
    copy ConvexList(G) to ConcaveList;

/* 2. Digging convex */
    for i=1 to the end of the ConcaveList
    {
        find the nearest inner point p_k ∈ G from the triangle (c_i1,c_i2,c_i3);
        // p_k should not closer to neighbor triangles of (c_i1,c_i2,c_i3) than
        // (c_i1,c_i2,c_i3)

        calculate eh= (D(c_i1,c_i2))+ D(c_i2,c_i3)+ D(c_i3,c_i1))/3;
        calculate dd=DD(p_k , {c_i1,c_i2,c_i3});

        if (eh/dd) > N                                  // digging process
        {
            insert new planes (c_i1,c_i2,k), (c_i2,c_i3,k), and (c_i3,c_i1,k)
                into the tail of ConcaveList;
            delete plane (c_i1,c_i2,c_i3) from the ConcaveList;

        }
    }

    Return ConcaveList;
```

As we can see, 2-dimensional concave hull algorithm can be simply extended to 3-dimensional algorithm. We also extend 3-dimensional algorithm to *n*-dimensional one ($n \geq 4$) following the same principle. We post *n*-dimensional concave hull program implemented using java to http://user.dankook.ac.kr/~bitl/dkuCH.

3. MEASURE OF CONCAVENESS

One of the goals of this study was to identify a method of expressing or visualizing the geometrical shape of the *n*-dimensional dataset. We believe that the concave hull is a useful concept to capture the geometrical shape. In this section, we propose the concaveness measure, which implies the degree of concaveness of a given dataset. We also introduce concaveness to the graph.

The degree of concaveness is calculated based on the difference between the convex hull and concave hull. In Fig. 9(a), the dark area between the convex hull and concave hull implies concaveness of the dataset. The concaveness measure *CM* is formulated as follows:

$$CM(c) = (ConcaveTotalLength(c) - ConvexTotalLength(c)) / ConvexTotalLength(c) \quad (4)$$

In equation (4), c is a target class of a dataset, $ConcaveTotalLength(c)$ produces the

total length of edges that form a concave hull on class c, and *ConvexTotalLength*(c) produces the total length of edges that form a convex hull on class c. The range of values of *CM* is ≥ 0. If $CM = 0$, the concave hull is exactly the same as the concave hull. If $CM(c_1) > CM(c_2)$, class c_1 has a higher concaveness than class c_2. Table 1 shows an example of *CM* for different shapes of classes in Fig. 8. The weak point of *CM* is that it simply shows the average degrees of concaveness; however, we cannot determine which areas are more concave than others. The concaveness graph shows more details regarding concaveness.

Table 1. Example of concaveness measure

Class	***CN*(*Ci*)**
C1	0.0223
C2	0.2242
C3	0.2079

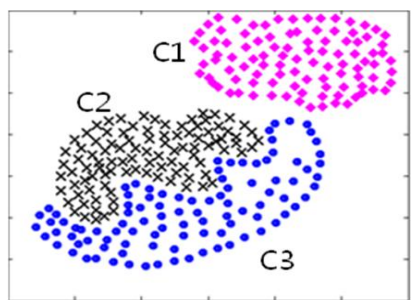


Fig. 8. Sample dataset for concaveness measure

The concaveness graph is a 2-dimensional graph that shows the degree of concaveness of each 'edge' of the concave hull. In the graph, the x axis contains the edges of the concave hull and the y axis expresses the degree of concaveness for each edge. The value y is calculated by

$$y = DE(k, convexEdge(k))$$
where $convexEdge(k)$ is corresponding convex edge to concave edge k (5)

Fig. 9 shows an example of corresponding convex edge to concave edges, and Fig. 10 shows an example of the concaveness graph. It is evident that the shape of the target class can be estimated from its concaveness graph. Therefore, the concaveness graph is a useful tool to analyze the shape of n-dimensional datasets.

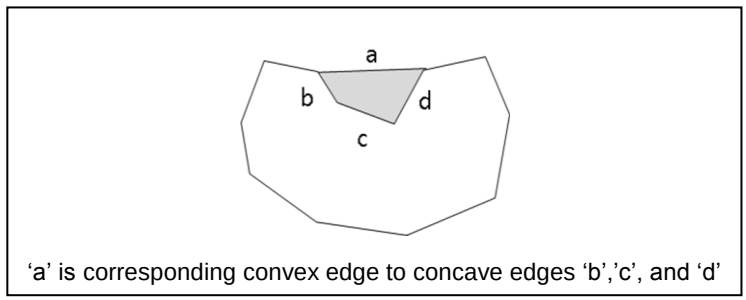
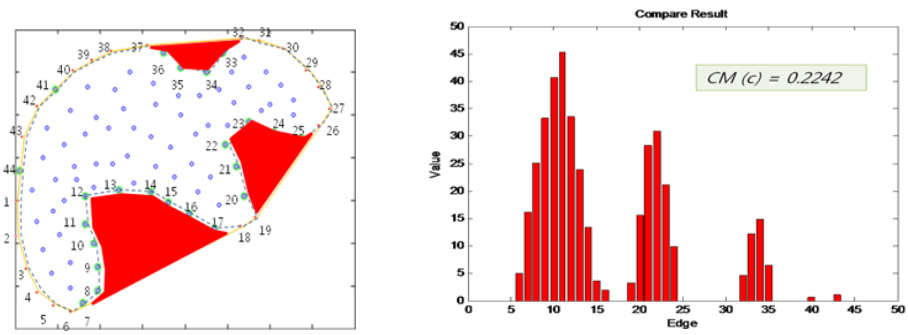


Fig. 9. An example of corresponding convex edge



(a) Target class *c* (b) Concaveness graph for class *c*

Fig. 10. Target class and its concaveness graph

4. DISCUSSION

4.1 Robustness of the proposed algorithm

Galton [24] suggested evaluation criteria for convex/concave hull algorithms. In this section, we describe the evaluation of the proposed algorithm following Galton's criteria. In his criteria, *S* is a set of points and *R*(*S*) is a set of boundary points of the concave hull of *S*.

(1) Should every member of *S* fall within *R*(*S*), as in Fig. 11(a,c–g), or are outliers permitted, as in Fig. 11(b)?

Outliers are not permitted in our proposed algorithm. The proposed algorithm starts from a convex hull of *S*, and every members of *S* falls within the convex hull. Each boundary points looks for the nearest inside points during digging process. If the nearest point *x* is excluded from the digging process, then point *y*, farther than *x*, is always excluded from the digging process. Therefore, there is no possibility that an outliers may exist.

(2) Should any points of *S* be allowed to fall on the boundary of *R*(*S*), as in Fig.

11(a–b,d–g), or must they all lie within its interior, as in Fig. 11(c)?

Any point of S is allowed to fall on the boundary of $R(S)$. In the proposed algorithm, a convex hull and concave hull are always formed by a point of S.

(3) Should $R(S)$ be topologically regular, as in Figure 5(a–c, e–g), or can it contain exposed point or line elements, as in Fig. 11(d)?

The concave hull is topologically regular in the proposed algorithm. In the algorithm, if a point x becomes a boundary point, no other boundary points can try dig to x. Therefore, there is no possibility that an exposed point or line element takes place. Fig. 11(d) is changed to Fig. 12 in the proposed algorithm.

(4) Should $R(S)$ be connected, as in Fig. 11(a–d,f,g), or can it have more than one component, as in Fig. 11(e)?

The proposed algorithm cannot produce more than one component, as in Fig. 11(e), for the same reason as in question (3).

(5) Should $R(S)$ be polygonal, as in Fig. 11(a–e,g), or can its boundary be curved, as in Fig. 11(f)?

The proposed algorithm cannot produce a curved boundary.

(6) Should $R(S)$ be simple, i.e., its boundary is a Jordan curve, as in Fig. 11(a–c,f), or can it have point connections as in Fig. 11(g)?

The proposed algorithm always produces a Jordan curved boundary,for the same reason, as in question (3).

(7) How big is the largest circular (or other specified) sub-region of $R(S)$ that contains no elements of S like Fig. 13?

In the proposed algorithm, the size of the circular sub-region that contains no elements of S depends on threshold N. If we choose big N, the concave hull may contain a large empty region, whereas small N produces a small empty region.

(8) How easily can the method used be generalized to three (or more) dimensions?

The proposed algorithm can be easily generalized to three (or more) dimensions. (*See* Algorithm 2)

(9) What is the computational complexity of the algorithm?

The time complexity of the proposed algorithm for 3-dimensional S is $O(n\log n + rn)$.

Our proposed algorithm satisfies Galton's evaluation criteria, with the exception of question (5). The question has no relationship to the robustness of the proposed algorithm.

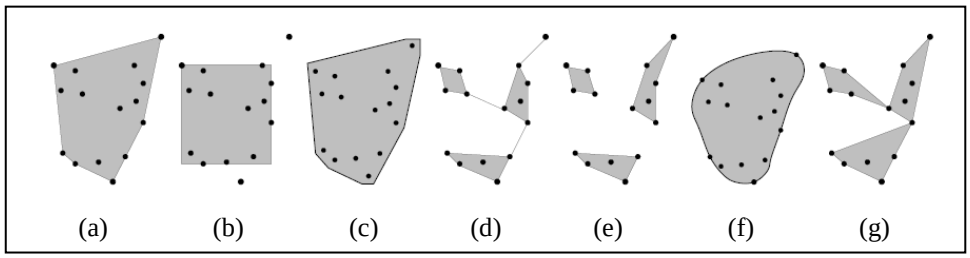


Fig. 11. Examples to illustrate the evaluation criteria

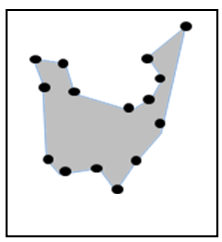


Fig.12 Changed shape for Fig. 11

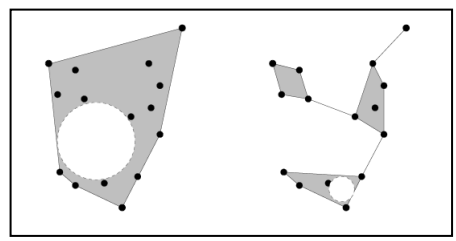


Fig. 13. Empty space as a criterion for region forming

4.2 Comparison with other concave hull algorithms

Before we compare proposed algorithm with others, we discuss about time complexity of our proposed algorithm. In our proposed concave hull algorithm, finding nearest inside points – these are candidates of target spots for digging – from boundary edges is a time-consuming process. Developing a more efficient method for this process is a future research topic.

Time complexity of proposed algorithm includes two parts:

– T1: generation of convex hull
– T2: digging of convex hull

Time complexity for T1 is depends on used algorithm. As a case of *QuickHhull* algorithm [23], approximate time complexity is known as $O(n \log n)$ for 2-dimensional dataset. T2 process contains several sub parts, and their time complexity of 2-dimensional algorithm is as follows:

– T2-1: finding nearest inner points for each convex edges: $O(rn)$
(r is a number of points in convex/concave hull list)
– T2-2: calculate decision distance: $O(r)$
– T2-3: add/delete edge from *concaveList*: $O(3)$

Therefore, total time complexity is $O(n \log n + rn)$. In the case of n-dimensional algorithm, approximate time complexity for T1 is known as $O(n(r^m/m!)/r)$ [23] where n is a size of dataset, r is a number of components in convex hull list, d is a dimension of dataset, and $m = \lfloor d/2 \rfloor$. Time complexity for T2 is as follows

– T2-1: finding nearest inner points for each convex edges: $O(rn)$
(r is a number of points in convex hull list)
– T2-2: calculate decision distance: $O(r)$
– T2-3: add/delete edge from *concaveList*: $O({}_dC_2)$

Total time complexity of proposed algorithm is $O(n(r^m/m!)/r + rn + {}_dC_2)$.

The number of points in the convex/concave hull list, r, depends on the dataset. If a dataset is d-dimensional and has n points (instances), minimum value of r is $d*(d+1)$ and maximum value is $d*n$. If every point of the dataset is located in the simplest region – for example, a triangle is the simplest region for a 2-dimensional dataset – r has a minimum value. If every point of a dataset is used to form boundary components, r has maximum value. If $r = d*n$, the time complexity of T2-1 is $O(dn^2)$ and total time complexity of the n-dimensional algorithm exceeds $O(dn^3)$. In general, if the dimension of a dataset is increased, the required number of points to form boundary components of the concave hull increases. For example, a 20-dimensional dataset needs 21 points for a boundary component whereas a 2-dimensional dataset needs only two points. Therefore, if the dimension of a dataset increases, r approaches $d*n$, and computation time increases rapidly.

Now we compare proposed algorithm with Swing Arm [24], KNN-based [26], and χ-Shape [25] algorithms following Galton's criteria. Table 2 summarizes the result. Proposed algorithm shows good time complexity and extensibility to higher dimension. Other algorithms are difficult or impossible to extend because of their limitations of foundational method. For example, χ-Shape makes Delaunay triangulation and erases some outside edges following threshold value. In that case, Delaunay triangulation can maintain digging points. In the case of 3-dimensional Delaunay triangulation, the shape of it is same as convex hull, and it does not maintain digging points inside of Delaunay triangulation. As a result, χ-Shape cannot extend to 3-dimensional algorithm.

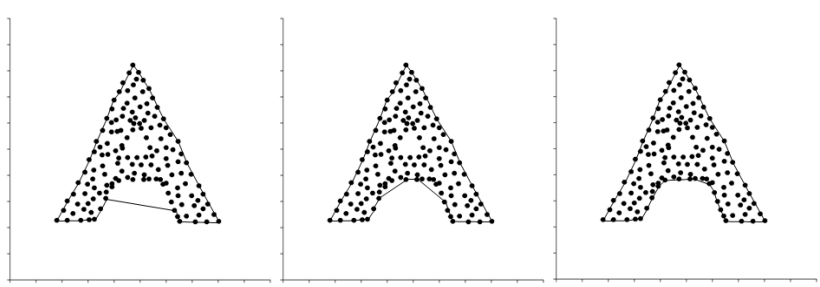
Table 2. Comparison of concave hull algorithms

	Swing Arm	**KNN-based**	**χ-Shape**	**Proposed**
Every point is inside of concave hull	Yes	Yes	Yes	Yes
Boundary of concave	Yes	Yes	Yes	Yes

hull is made by given points				
Concave hull includes line region	Yes	No	No	No
Concave hull can be separated to multiple regions	Yes	No	No	No
Produce curve boundary	No	No	No	No
Boundary is a Jordan curve	Yes	No	Yes	Yes
Has large empty space inside of concave hull	Depends on length of swing arm	Depends on parameter	No	Depends on threshold
Possibility of extension to higher dimension	Difficult	Difficult	Impossible	Easy
Time complexity*	$O(n^3)$	$O(n^3)$	$O(n\log n)$	$O(n\log n + rn)$

(* described time complexity is case of 2-dimensional algorithm)

Fig. 14 shows 2-dimensional shapes of concave hulls derived from the χ-Shape and the proposed algorithms. χ-Shape has a parameter 'Edge Removal Length (EDL)', like the threshold of the proposed algorithm that determines the shape of the concave hull. The change of shape following the threshold differs from the χ-Shape following EDL due to the difference of forming algorithms. In general, a simple shape such as 'A' differs slightly between the χ-Shape and proposed algorithms. In a complex case, such as 'S', χ-Shape produces smoother shaped boundary than does the proposed algorithm. However χ-Shape cannot extend to 3 or higher dimensional data.



χ-Shape

ERL=50 ERL=30 ERL=10

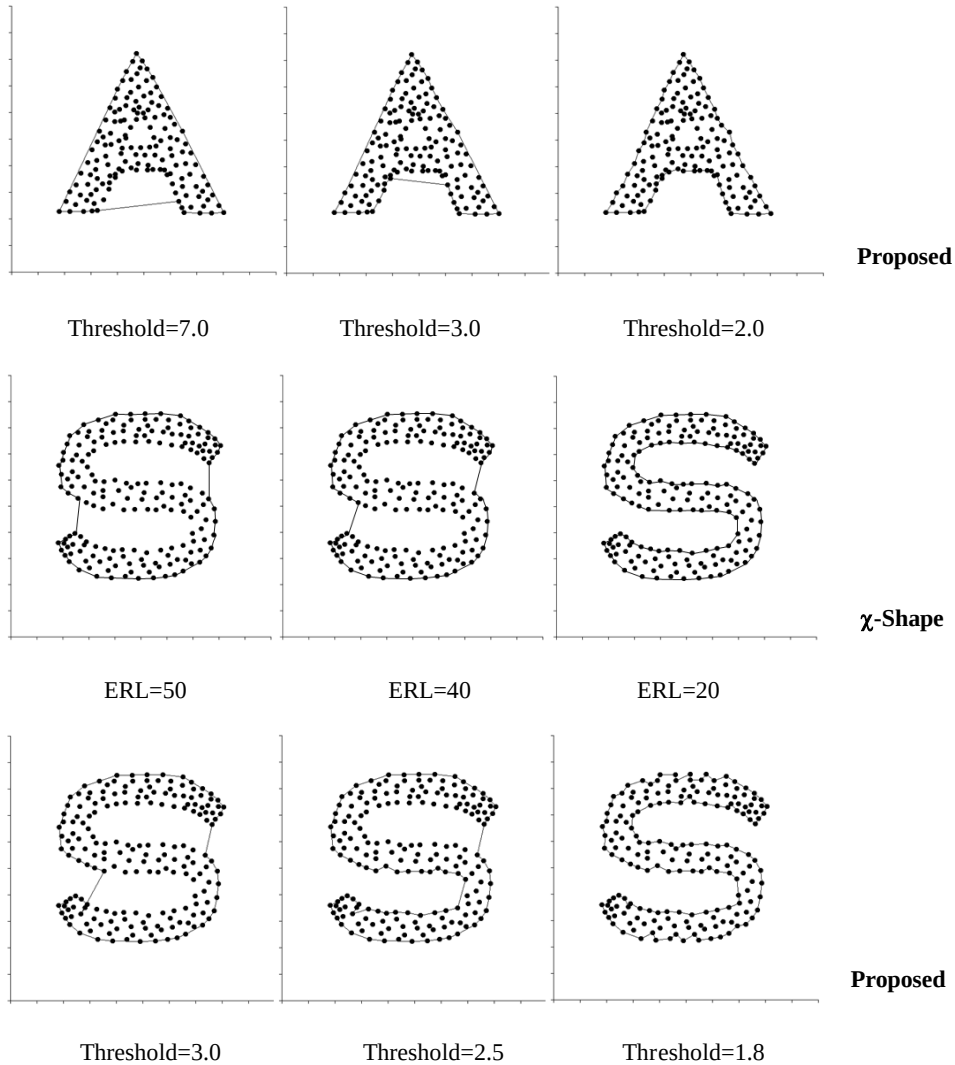


Fig. 14. 2-D Concave hulls derived from χ-Shape and proposed algorithms

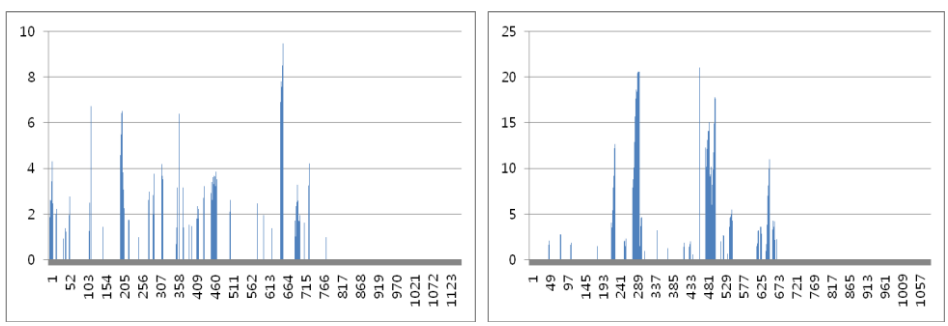
4.3 Experiment of concaveness measure

We experiment with the concaveness graph and concaveness measure on two real datasets. Table 3 summarizes the two datasets. We take the datasets from UCI machine learning repository (http://archive.ics.uci.edu/ml/). We use threshold = 2.0. Fig. 15 and Fig. 16 show the experimental results. We can see each dataset class has different levels of concaveness and concave location. The main advantage of the proposed concaveness measure and graph is that they can visualize the geometric shape of an *n*-dimensional class in a dataset. This visualized information can be used for feature selection [1-6] or a classification task, because concaveness of a class has a relationship with classification

accuracy. As shown in Fig. 18, both datasets A and B look strongly separable, but dataset B may provide lower classification accuracy than dataset A, because the geometric shape of dataset B is more complex and mixed than that of dataset A. Most related research on dataset analysis focus on separability among classes. In contrast, few studies have considered the geometrical shape of classes during the classification task. The concern of topology is supposed to be increased in the classification task, and the concaveness measure/graph may contribute to the work.

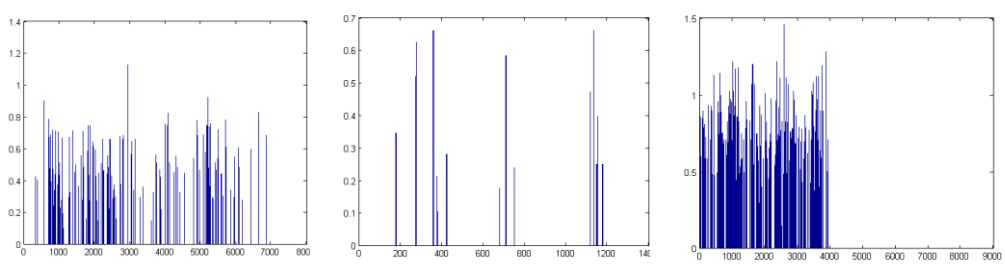
Table 3. Summary of two datasets

Dataset	No. of instances	No. of classes	No. of features
Mammographic	961	2	6
Car Evaluation	1728	3	6
Abalone	4177	29	8



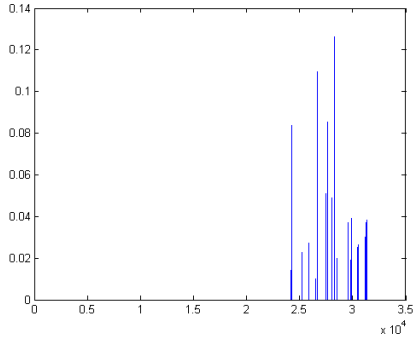
Benign (CM = 0.3009) Malignant (CM=0.2921)

Fig.15. Concaveness measure/graph for Mammographic Mass dataset

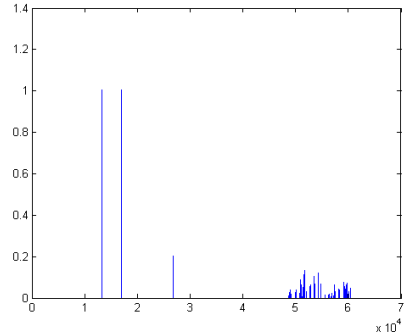


Acc (CM=0.1370) Good (CM=0.0592) Unacc (CM=0.5594)

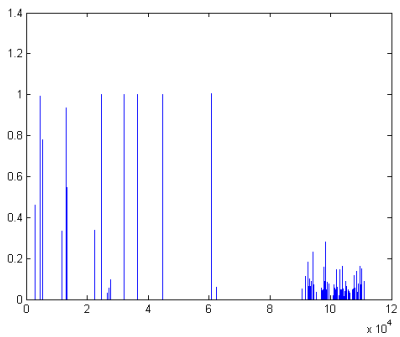
Fig.16. Concaveness measure/graph for Car Evaluation dataset



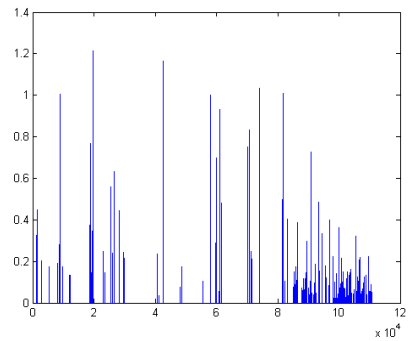
Class 5 (CM= 0.0009)



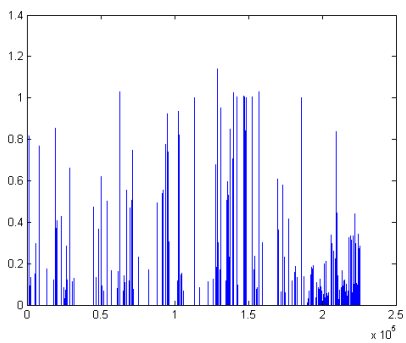
Class 6 (CM=0.0023)



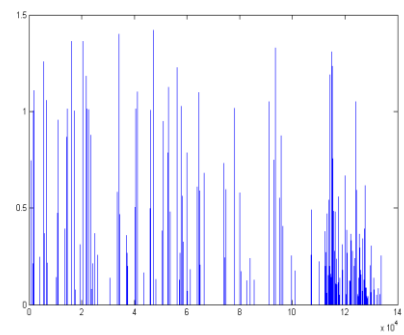
Class 7 (CM=0.0037)



Class 8 (CM=0.0111)

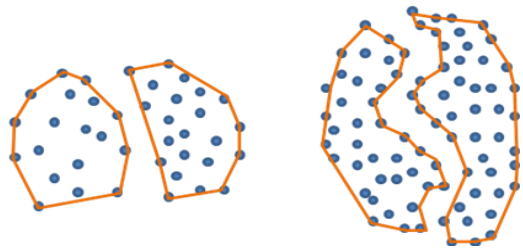


Class 9 (CM=0.0082)



Class 10 (CM=0.0150)

Fig. 17. Concavness measure/graph for Abalone dataset (for selected classes)



(a) dataset A (b) dataset B

Fig. 18. Two datasets that have different geometrical shape

5. CONCLUSION

A concave hull can more precisely capture the geometric boundary of a dataset than a convex hull. Here, we propose a new algorithm to produce a concave hull that is easy to understand and implement. The application area of the concave hull is very wide. Convex hulls are already widely used in geographic information processing, image processing, pattern recognition, and feature selection in machine learning areas. If the convex hull is substituted with a concave hull in those tasks, increased performance or accuracy can be expected. Our contribution point is that we propose concave hull algorithm for *n*-dimensional dataset whereas previous researches suggest for 2-dimensional datasets.

Concaveness measure and graph is one of application of concave hull. In the classification task, analysis of the dataset is important, but only basic statistical information can be obtained from a dataset if it is highly dimensional. If a concaveness graph is used, then information of geometric boundary can be obtained, and this information can be applied for planning classification tasks.

We implement concave hull algorithm and concaveness measure/graph using java, and post them to the website http://user.dankook.ac.kr/~bitl/dkuCH.

REFERENCES

1. P. M. Neil, S. Qiang and R. Jensen, "Distance measure assisted rough set feature selection," *Proceedings of the 16th International Conference on Fuzzy Systems*, 2007, pp. 1084-1089.
2. A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, Vol. 97(1-2), 2007, pp. 245-271.
3. R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, Vol. 97(1-2), 1997, pp. 273-324.
4. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, Vol. 3, 2003, pp. 1157-1182.
5. W. S. Meisel, "Computer-oriented approaches to pattern recognition," *Academic Press*, 1997.

6. Dash M. and Liu H., "Feature selection for classification," *Intelligent Data Analysis*, 1997, pp. 131-156.
7. J. Qing, H. Huo and T. Fang, "Nearest convex hull classifiers for remote sensing classification," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 37, 2008, pp. 589-594.
8. T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to algorithms, second edition," *The Messachusetts Institute of Technology press*, 2001.
9. X. Kong, E. H. and G. Toussain, "The graham scan triangulates simple polygons," *Pattern Recognition Letters*,Vol. 11, 1990, pp. 713-716.
10. Yang X. and Wang G, "Accelerating algorithm for 3D convex hulls construction," *Journal of Zhehiang University*, Vol. 33, 1999, pp. 111-114.
11. Cinque L. and Maggio C., "A BSP realization of Jarvis algorithm," *10th International Conference on Image Analysis and Processing*, 1999, pp. 247.
12. B. Valentina. "Survey of algorithms for the convex hull problem," *Oregon State University*, 1999.
13. Guo F., Wang X. Z. and Li Y., "A new algorithm for solving convex hull problem and its application to feature selection," *Proceedings of the Seventh International Conference on Machine Learning and Cybernetics*, 2008, pp. 369-373.
14. S. Kokichi. "Robust gift wrapping for the three-dimensional convex hull," *Journal of Computer and System Sciences*, Vol. 49(2), 1994, pp. 391-407.
15. B. Chazelle. "An optimal convex hull algorithm in any fixed dimension," *Discrete and Computational geometry*, Vol. 10(1), 1993, pp. 377-409.
16. P. Szczypinski and A. Klepaczko. "Convex Hull-Based Feature Selection in Application to Classification of Wireless Capsule Endoscopic Images," *Advanced Concepts for Intelligent Vision System, 11th International Conference, ACIVS 2009*, 2009, pp. 664-675.
17. L. Zhou, K. K. Lai and J. Yen, "A New Approach with Convex Hull to Measure Classification Complexity of Credit Scoring Database," *Business Intelligence: Artificial Intelligence in Business, Industry and Engineering, Proceedings of the second international conference on business intelligence and financial engineering*, 2009, pp. 441-444.
18. X. Zhou and Y. Shi. "Nearest Neighbor Convex Hull Classification Method for Face Recognition," *Proceeding of the 9th international conference on computational science*, 2009, pp. 570-577.
19. M. Stout, J. Bacardit and J. D. Hirst, "Prediction of recursive convex hull class assignments for protein residues," *Bioinformatics*, Vol. 24(7), 2009, pp. 916-923.
20. K. Lu and T. Pavlidis. "Detecting textured objects using convex hull," *Machine Vision and Applications*, Vol. 18(2), 2007, pp. 123-133.
21. B. Yuan and C. L. Tan, "Convex hull based skew estimation," *Pattern Recognition Letters*, Vol. 40(2), 2007, pp. 456-475.
22. Minhas R. and Wu J., "Invariant feature set in convex hull for fast image registration," *Systems, Man and Cybernetics, IEEE International Conference on*, 2007, pp. 1557-1561.
23. Barber C.B., Dobkin D.P. and Huhdanpaa H. "The Quickhull Algorithm for Convex Hull," *ACM Transactions on Mathematical Software*, Vol. 22, 1996, pp. 469–483.
24. A. Galton and M. Duckham, "What is the Region Occupied by a Set of Points?,"

GIScience 2006, LNCS 4197, 2006, pp. 81-98.
25. M. Duckham, L. Kulik, M. Worboys and A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane," *Pattern Recognition* , Vol. 41, 2008, pp. 3224-3236.
26. A. J. C. Moreira and Y. M. Santos, "Concave hull: A k-nearest neighbors approach for the computation of the region occupied by a set of points," *Proceedings of the International conference on computer Graphics Theory and Applications*, 2007, pp. 61-68.

Jin-Seo Park is a received his Bachelor degree in computer science from Dankook University, Korea, in 2009. He is currently M.S. student in department of NanoBioMedical Science at Dankook university. He is also researcher of WCU Research Center of NanoBioMedical Science. His main research interests are machn learning algorithms and bioinformatics. He developed *n*-dimensional concave hull algorithm.

Se-Jong Oh received a Doctor, Master, and Bachelor degree in Computer Science from Sogang University, Korea, in 2001, 1991, and 1989. From 2001 to 2003, he was a postdoctoral fellow in the laboratory for Information Security Technology at George Mason University, USA. Since 2003 he joined the Department of Computer Science at Dankook University, Korea, and is currently associate professor in WCU Research Center of NanoBioMedical Science. His main research interests are bioinformatics, information system, and information system security.