ABSTRACT
This paper proposes a new algorithm for identifying patterns within data, based on data depth. Such a clustering analysis has an enormous potential to discover previously unknown insights from existing data sets. Many clustering algorithms already exist for this purpose. However, most algorithms are not affine invariant. Therefore, they must operate with different parameters after the data sets are rotated, scaled, or translated. Further, most clustering algorithms, based on Euclidean distance, can be sensitive to noises because they have no global perspective. Parameter selection also significantly affects the clustering results of each algorithm. Unlike many existing clustering algorithms, the proposed algorithm, called data depth based clustering analysis (DBCA), is able to detect coherent clusters after the data sets are affine transformed without changing a parameter. It is also robust to noises because using data depth can measure centrality and outlyingness of the underlying data. Further, it can generate relatively stable clusters by varying the parameter. The experimental comparison with the leading state-of-the-art alternatives demonstrates that the proposed algorithm outperforms DBSCAN and HDBSCAN in terms of affine invariance, and exceeds or matches the robustness to noises of DBSCAN or HDBSCAN. The robustness to parameter selection is also demonstrated through the case study of clustering twitter data.

CCS Concepts
• Computing methodologies → Machine learning algorithms; • Theory of computation → Design and analysis of algorithms; • Mathematics of computing → Non-parametric statistics;

Keywords
Cluster analysis; data depth; density-based clustering analysis; affine invariant clustering

1. INTRODUCTION
The main contribution of this paper is to propose and investigate a new algorithm for clustering data, based on data depth. Clustering analysis is a common technique to extract actionable information from the underlying data. There are a variety of applications based on clustering analysis: Customer segmentation [9], image recognition [30], genetic sequencing [11] or human mobility patterns [19].

The most commonly used clustering method is density-based clustering in spatial data such as DBSCAN [12]. This method is based on density measurements by using the maximum radius of the neighbors (eps) and the minimum number of points (MinPts) within the radius of the neighbors. DBSCAN can not only detect arbitrary shaped clusters, but also requires no priori knowledge about the data set (i.e., unsupervised learning).

However, it is difficult to identify coherent clusters from complex data by using a threshold such as distance or count. Further, if the data is simply projected into another coordinate system or if we cluster feature points in the images taken by Unmanned Aircraft System (UAS), it must determine appropriate parameters again because a density-based approach is not affine invariant. In addition, a density-based approach can be sensitive to noises. It does not measure centrality or outlyingness because there is no global perspective in its interpretation.

This paper therefore provides a new algorithm, called data depth based clustering analysis (DBCA). Data depth measures the centrality of a data point with respect to a given data set [32]. For example, Figure 1 presents the Mahalonobis depth contours over 90 randomly distributed points — Section 3.1 explains the Mahalonobis data depth. A depth function orders data by their degree of centrality. High
ing applies a statistical approach to detect cohesive groups by using a formal model such as a Gaussian mixture model [13, 7]. The model-based clustering determines the most likely number of clusters as an initial parameter by a user. Further, k-means clustering cannot guarantee it will discover the optimal set of clusters. Thus, one k-means refinement is to minimize the squared Euclidean distances within clusters and maximizing the differences between the center by minimizing the squared Euclidean distances within clusters and maximizing the differences between the center and other clusters.

In the remainder of this paper, we review our algorithm's characteristics in the context of related work (Section 2). Then we present data depth and the algorithm itself (Section 3). Section 4 presents an experimental evaluation of the algorithm in terms of affine invariance, the robustness to noises, and the robustness to parameter selection. Finally, Section 5 concludes the paper with a discussion of the limitations of the approach, and future work.

2. BACKGROUND

Research into clustering analysis is already well advanced. There are at least five broad approaches to clustering algorithms in the literature: hierarchical clustering, k-means clustering, model-based clustering, graph-based clustering, and density-based clustering.

First, hierarchical clustering builds a hierarchical structure of data [17, 27]. A dendrogram presents the hierarchy of clusters. However, it is hard to apply the hierarchical clustering on big data due to high computational complexity. Further, it is necessary to define the dendrogram cut manually.

K-means clustering is based on simple principles such as squared error [15, 3]. For example, the algorithm randomly creates k partitions and calculates the centroid of each partition. Next, the algorithm assigns each object to the cluster center by minimizing the squared Euclidean distances within clusters and maximizing the differences between the clusters. Then the algorithm repeats this process until the within-cluster variation improves. Thus, one k-means result can be greatly different with another one. Therefore, k-means clustering cannot guarantee it will discover the optimal set of clusters. Further, the algorithm requires the number of clusters as an initial parameter by a user.

While the methods mentioned as above are based on a heuristic approach, model-based clustering determines the most likely number of clusters using a formal model such as a Gaussian mixture model [13, 7]. The model-based clustering applies a statistical approach to detect cohesive groups of data. However, the probability model suffers from non-Gaussian distributions of the underlying data because the model-based clustering assumes that data is independent and identically distributed. Moreover, it is difficult to identify arbitrary shape of data by using the normal distribution assumption of the underlying data.

Another approach is a graph-based analysis [10, 35]. The shape of a set of points is characterized by convex hull based on the Delaunay triangulation. This approach, however, cannot identify the line or curve shape of data as a cluster.

Therefore, density-based clustering algorithms have been broadly used for understanding and identifying patterns in spatial data because a density-based approach can detect arbitrary shaped clusters. One of the most famous methods is DBSCAN [11]. Since this paper has been published, there is a wide variation in different density datasets [2, 4], improved computational complexity [28, 6], and appropriate parameter selection [20]. Further, [14, 8] provide a hybrid method with hierarchical clustering approach, which results in improving the robustness to noises. [5] discovers spatial-temporal patterns in data, based on DBSCAN.

However, all the approaches discussed above are not affine invariant. Affine invariance is often considered as a basic requirement in multivariate statistical inference [24]. Although all these approaches are useful techniques for understanding the structure of the underlying data, they tend to be subjective because the results can be different when the underlying data is rotated, scaled, or translated. Thus, it can be a heuristic and time consuming process to detect the same cluster by changing parameters.

Therefore, this paper provides an affine invariant clustering method based on data depth. Data depth is nonparametric inference for multivariate data. Statistical depth functions are different with model-based clustering methods because statistical depth functions do not rely on the assumption of data distribution. A couple of papers have already utilized statistical depth functions for data classification [22] and clustering [18]. In particular, [18] applies data depth to k-means clustering. This approach, however, is not well-adapted for use in spatial data clustering because k-means based clustering is not designed for detecting arbitrary shapes of data.

Consequently, this paper presents and tests a new clustering algorithm based on data depth. Our algorithm is affine invariant. Further, it can detect arbitrary shaped data with improved robustness to noises.

3. ALGORITHM

3.1 Data depth

Data depth measures how deeply a point is located with respect to given data sets. [33] originally proposed a “half-space” depth in order to present information about multivariate data distribution, based on center-outward ordering. Similarly, there are several statistical depth functions as multivariate location estimators such as simplicial depth, projection depth, zonoid depth or Mahalanobis depth [36, 32, 26]. This study uses a Mahalanobis depth function to measure the centrality of a point within a cloud of data because of its fast and easy computability.

The Mahalanobis depth function can be defined as below:
The center-outward ordering of the Mahalanobis depth function presents that the deepest points near the center should have the higher depth and the lower depth is related to its outlyingness. However, data depth presents globally maximizing depth. Thus, it is difficult to present multimodality features of the underlying data. Thus, Equation 1 can be modified as below:

\[ M_d(x|X) = [1 + (x - \bar{X})^\prime M^{-1}(x - \bar{X})]^{-1} \]

where \( M \) is the covariance matrix of \( X \). This study uses a robust minimum volume ellipsoid (MVE) estimator for the covariance matrix. Let \( M \) be a robust minimum volume ellipsoid estimator for the covariance matrix. Therefore, each point \( x_i \) can be regarded as a center point so that it is possible to calculate data depth from each point with respect to a given data set.

3.2 Algorithm

This paper defines some concepts for the formulation of DBCA. The definitions follow the style of DBSCAN [11]. However, our approach relies on data depth, rather than density.

Let \( X = \{x_1,...,x_n\} \) be a finite set of two-dimensional points in \( \mathbb{R}^2 \) and a depth threshold, \( \theta \in [0,1] \). In general, it is reasonable to determine \( \theta \in [0.5,1] \) because \( \theta < 0.5 \) is related to outlyingness.

Definition 1. Core neighbor: A point \( x_j \) is called a core neighbor of \( x_i \) if \( M_d(x_j|x_i) > \theta \), where \( \theta \) is a depth threshold that regards the point \( x_j \) as a closer neighbor to \( x_i \).

Definition 2. Depth-connection: A point \( x_k \) is a depth-connected to an point \( x_i \) if there exists a chain of points, \( p_1,p_2,...,p_n \), based on a core neighbor, where \( p_1 = x_i \) and \( p_n = x_k \).

Definition 3. Depth-based cluster: A cluster \( C \) is a non-empty maximal and depth-based subset of data sets \( X \) such that \( C = \{C_1,...,C_k\} \), where \( k \leq n \) and each cluster \( (C_i \in C) \) is depth-connected.

Figure 2 presents the concept of depth-connection (Definition 2). Although the point \( x_{50} \) is not a core neighbor of the point \( x_1 \) in Figure 2a, two points are depth-connected by a continuous chain of a core neighbor starting from the point \( x_1 \). For example, \( x_1 \)'s core neighbors are \( x_2 \) and \( x_3 \), where a depth threshold is \( \theta = 0.8 \) in Figure 2b. Then we repeat this process from \( x_2 \) and \( x_3 \) until all depth-connected points are identified.

Based on these definitions, Algorithm 1 provides the protocol for identification of depth-based clusters. In brief, Algorithm 1 can be summarized as follows:

- The algorithm calculates the Mahalanobis depth of each point. However, this process can be materialized to reduce time complexity (Algorithm 1, line 4).
- The algorithm starts data depth based clustering from a point \( x_1 \) (Algorithm 1, line 5) and retrieves core neighbors from \( x_1 \) with a depth threshold \( \theta \) (Algorithm 1, line 7).
- If there are core neighbors from \( x_1 \), a new cluster number is assigned (Algorithm 1, line 9) and a chain of points, based on depth-connection (Definition 2), is constructed (Algorithm 1, line 11). The same cluster number is assigned to the chain of points.
- If there are no core neighbors from \( x_1 \), the algorithm moves next point. This point can be regarded as a noise (outlier).
- Continue the process as above with other points until all of the points have been processed.

Some applications may require the minimal number of points within a cluster. It is not necessary to set up the minimal number of points as a parameter in DBCA. However, we can set up the minimal number of points optionally, based on the requirement of applications. For example, when we check the core neighbors of each point (Algorithm 1, line 8), it is possible to filter out the clusters based on the minimal number of points.

In terms of computational complexity, the matrix of Mahalanobis depth can be provided as input instead of \( X \), similar to the distance matrix of DBSCAN. Thus, it requires
Algorithm 1 Data depth based clustering algorithm

1: Input: A finite set of two-dimensional points, \( X = \{x_1, \ldots, x_n\} \) in \( \mathbb{R}^2 \) and a depth threshold \( \theta \in [0, 1] \)
2: Local variable: list of cluster number, \( Cl \), initialized empty; list of core neighbors, \( CNei \), initialized empty; cluster number, \( cn \), initialized 0
3: Output: Cluster number of each point, \( Cl \)
4: Compute Mahalanobis depth of each point in \( X \)
5: for \( i = 1 \) to \( n \) do
6: if \( CNei[i] = \emptyset \) then
7: Compute core neighbors (\( CNei[i] \)) of a point (\( x_i \)) using \( \theta \)
8: if \( CNei[i] \neq \emptyset \) then
9: set \( Cl[i] := cn + 1 \)
10: while \( CNei[i] \) do
11: Construct a chain of depth-connected

\( \mathcal{O}(n^2) \) memory, where \( n \) is the number of input points. The
time complexity therefore is mainly influenced by finding
core neighbors based on Mahalanobis depth. Such a query
is executed once by each point. With an indexing structure
such as \( R^* \)-tree, finding core neighbors from a point requires
\( \mathcal{O}(\log n) \). Thus, the time complexity of DBCA is \( \mathcal{O}(n \log n) \).

4. EXPERIMENTS

The algorithm described in the previous section was evaluated
regarding three key features: affine invariance, robustness to noises, and robustness to parameter selection.
For benchmarking, the performance of DBCA was compared
with the state-of-the-art two alternatives such as DBSCAN
[12] and HDBSCAN [8].

4.1 Experiment setup

This paper used sample clustering data from the scikit-learn [31] and the mlbench [23]. The data set is composed of
four data structures: circles, spirals, smiles, and shapes in
two dimension (see Figure 3a). This data set can be regarded as “ground truth” and its total number of points is
3,000.

Based on the original data set, this study made 10 data
sets for affine invariant tests including the original data. For
example, the original data was rotated, scaled, or translated.
Figure 3b presents one of affine invariant data examples such
as counterclockwise 45° rotation and 5 times X-axis expansion.
Similarly, this study generated 10 data sets for verifying
the robustness to noises. The study expanded the
original data in X and Y-axis direction and increased noises
(outliers) ranging from 0% to 50% (e.g., 0%, 5%, ..., 45%, 50%). The outliers were randomly generated from the mixture
of normal distributions outside the ground truth data set.
Some noises can locate near the ground truth data set.
These points are considered as outliers.

In terms of evaluation metric, this paper uses Adjusted
Rand Index (ARI) [16]. The ARI measures the similarity of
two clustering results based on the ground truth classification.
The class label of each object was calculated based on the
original data. The range of ARI is \(-1 \) to \( 1 \). The
ARI 1.0 indicates the perfect matching score, while negative
values and 0 indicate poor matching scores. Further, this study reported variation of information (VI) index [25], which
measures the amount of information lost and gained by changing
from clustering \( C_1 \) to clustering \( C_2 \). Thus, VI index should be zero when two clusters are the same, \( VI(C_1, C_2) = 0 \) if
and only if \( C_1 = C_2 \).

4.2 Affine Invariance

The first experiment investigated the performance of algorithms with respect to affine invariance. The algorithms were examined on 10 data sets. Each algorithm determined optimal parameters based on the initial data set. For example, DBCA used 0.9 as \( \theta \) threshold, DBSCAN determined 0.14 as \( eps \) and 2 as \( MinPts \), and HDBSCAN used 5 as a minimal cluster size. DBCA and DBSCAN identified one point as a noise (outlier) around the shape of Gaussian in the initial
data set. HDBSCAN, however, regarded this point as a normal
point. Three algorithms classified all points as the same
label except this point.

Based on these initial parameters, the study evaluated the
performance of algorithms. Figure 4 presents one of results
for affine invariant tests after the original data set has been
scaled by 5 times to X-axis and rotated by counterclockwise
45°. Our algorithm, DBCA, identified all shape of data perfectly. However, DBSCAN and HDBSCAN failed to generate
coherent clusters after the data set had been scaled and rotated.

Figure 5a presents the whole results of experiments measuring the ARI of each algorithm. As expected from Figure
4, DBCA generated coherent clusters, irrespective of affine transformation. However, DBSCAN and HDBSCAN presented poor performance. Interestingly, HDBSCAN presented
it could identify coherent clustering when X and Y-
axis are scaled by 5 times to X-axis and rotated by counterclockwise
45°. Our algorithm, DBCA, identified all shape of data perfectly. However, DBSCAN and HDBSCAN failed to generate coherent clusters after the data set had been scaled and rotated.

Figure 5b presents the VI index. There were similar
patterns and the same statistical results, compared
with ARI. Our DBCA, therefore, outperformed other approaches statistically and practically in terms of affine invariance.

Further, this study investigated the feasibility of affine
invariance with a real data set. Figure 6 presents two UAV
images from OpenDroneMap benchmark dataset [29]. Figure
6b is generated by clockwise 10° rotation and \( 0.8 \times \)
X-axis scale of Figure 6a. Then we can detect the same features from two images using AKAZE feature detection [1].
The identified features (key points) are the same although
their scale and locations are different in Figures 6c and 6d.
This study then applied clustering algorithms (DBCA,
DBSCAN, and HDBSCAN) to the identified features. The results of clustering are shown in Figure 7. DBCA used 0.9 as \( \theta \) threshold, DBSCAN applied 42 as \( eps \) and 5 as \( MinPts \), and HDBSCAN used 5 as a minimal cluster size. Each clus-
ter was assigned to a random color. Noises are represented as the small black circle. In order to compare the coherence of two clusterings, the ARI and VI index were calculated between clusters for each algorithm. The ARI of DBCA is 1, DBSCAN 0.63, and HDBSCAN 0.68. The VI index of DBCA is 0, DBSCAN 0.65, and HDBSCAN 0.78 respectively. Our approach, DBCA presented greater coherence although the image was scaled and rotated. This result is
4.3 Robustness to noises

The accuracy of DBCA, DBSCAN, and HDBSCAN was compared with different noises to signal ratio. The initial data set was scaled to ensure space where noises (outliers) can locate without significantly cluttering the shape of data. Our DBCA continuously used 0.9 as $\theta$ threshold. However, DBSCAN and HDBSCAN changed their optimal parameters: 1.5 as $\epsilon$ and 2 as $MinPts$, and 6 as a minimal cluster size respectively.

Figure 8 presents the clustering performance of each algorithm over 10% noises. In spite of 10% noises, DBSCAN failed to differentiate spirals data in Figure 8b.

As the noise to signal ratio increases, the performance of all algorithms’ ARI is presented in Figure 9a. The performance of DBSCAN was dramatically weakened after 20% noises. This visual impression was confirmed by statistical analyses. Welch’s F and robust ANOVA test indicated that there was significant difference in ARI, $p < 0.05$ at $\alpha = 0.05$. The robust post hoc tests revealed that there was significant difference between DBSCAN and DBCA, and DBSCAN and HDBSCAN, $p < 0.05$. However, there was no significant difference between DBCA and HDBSCAN, $p = 0.85$. This result was repeated in terms of effect size: $\xi = 0.84$ between DBCA and DBSCAN, $\xi = 0.81$ between DBSCAN and HDBSCAN, and $\xi = 0.19$ between DBCA and HDBSCAN. The effect size of DBCA and HDBSCAN is small effect size. Conversely, other comparisons presented large effect size. Thus, there are statically and practically significant difference between DBSCAN and other approaches. However, there was no significant difference between DBCA and HDBSCAN in terms of the robustness to noises. The same results were obtained from the VI index in Figure 9b.

4.4 Robustness to parameter selection

It is important to generate relatively coherent clusters by varying parameters of each clustering algorithm. A slight change of parameters can lead to significantly different clusters. Selecting the correct parameters may be difficult to achieve automatically. Thus, this paper investigated how each algorithm responded to the change of parameters.

This study used Twitter data from the Chicago area, 2014. Three users were randomly selected. One of them tweeted the most (i.e., 2,682 tweets) during 2014. Others had 1,000 and 500 tweets respectively. Then we can apply clustering algorithms to identify their main activity areas.

In order to compare the robustness to parameter selection, we compared the ARI value of an initial parameter with that of each varied parameter. For example, we increased a depth
threshold from 0.5 to 0.95 for DBCA. The $\epsilon$ was decreased from 2,500m to 50m for DBSCAN with the same $MinPts$, 5. The minimal cluster size of HDBSCAN was reduced from 11 to 2. The range of parameters can not be comparable due to the different unit of each algorithm. Further, this experiment is not meant to evaluate the correctness of clusters. It is meant to present how each algorithm performs by varying parameters.

Figure 10 presents the robustness to parameter selection. As we can see, there are no stable patterns for DBSCAN and HDBSCAN, which means the clusters can be different due to parameters. However, DBCA generated relatively stable changes. Thus we can get some predictable results by varying a parameter in DBCA. Interestingly, Welch’s F test revealed that there was no significant difference among twitter users for DBCA, $p = 0.847$. However, there was significant difference among twitter users for DBSCAN and HDBSCAN, $p < 0.05$. 

Figure 7: Comparison of clustering performance for feature points
5. DISCUSSION AND CONCLUSIONS

This paper has presented that DBCA is not only affine invariant, but also robust to noises and to parameter selection. Our algorithm is simple and comparably efficient in the sense that DBCA requires one intuitive parameter such as a depth threshold, $\theta$. Its time complexity is $O(n \log n)$,
which is comparable with that of DBSCAN and better than HDDBSCAN, $O(n^2)$.

Affine invariance can be treated effectively using data depth. This study shows how classic clustering algorithms fail to be adapted to the affine invariant cases. DBCA, however, generates coherent clusters, irrespective of affine transformation.

In terms of the robustness to noises, DBCA improves or achieves the results of DBSCAN and HDDBSCAN. Data depth can measure the centrality and outlyingness of the underlying data such that DBCA is able to deal with noises, based on center-outward ordering. HDDBSCAN similarly demonstrates the comparable robustness to noises. While HDDBSCAN must take a couple of steps (e.g., build the minimum spanning tree of the distance weighted graph or construct a cluster hierarchy of connected components) to extract the stable clusters from noises, our DBCA can operate within just one step process.

In addition, the experiment demonstrates that DBCA is robust to parameter selection. When we use complicated data, it is hard to select an appropriate parameter for grouping data. By varying the parameters, DBCA can get some predictable results. However, DBSCAN and HDDBSCAN cannot find stable patterns such that it is difficult to set up the parameter automatically.

Even though DBCA only requires one intuitive parameter, this paper did not provide an adaptive parameter scheme based on data sets. If there are very different density data, the current global depth threshold cannot detect all patterns from them. We are expanding the current algorithm so that it can automatically optimize the parameter based on data densities without a manual selection process.

In addition, this paper uses the Mahalanobis depth to calculate the center-outward ordering. However, the Mahalanobis distance is not robust when there is a heavy tail distribution in the underlying data. There are alternative robust statistical depth functions such as projection depth. However, they require high computational complexity. It is necessary to investigate alternative depth functions with better computational complexity.

Our approach can have many applications in computer vision and image processing systems. We plan to expand DBCA to conduct feature matching for the homogeneous images of UAS in agriculture areas. Using cluster matching, we can constrain the feature matching within the cluster instead of whole image searching, which results in the improvement of efficiency and accuracy.

6. ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation (grant numbers: 0846655, 1047916, and 1443080) and the National Nuclear Security Agency of the Department of Energy through the Consortium for Nonproliferation Enabling Capabilities (grant number: DE-NA0002576). Further, the work used the ROGER supercomputer, which is supported by NSF under grant number: 1429699.

7. REFERENCES


